

REPORT DOCUMENTATION PAGE			Form Approved OMB NO. 0704-0188		
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA, 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 15-08-2008		2. REPORT TYPE Final Report		3. DATES COVERED (From - To) 1-Aug-2005 - 31-Jul-2008	
4. TITLE AND SUBTITLE Mathematical Frameworks for Diagnostics, Prognostics and Condition Based Maintenance Problems (W911NF-05-1-0426)			5a. CONTRACT NUMBER W911NF-05-1-0426		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER 611102		
6. AUTHORS Andrew Scott, Kaveh Heidary			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES Alabama A&M University Office of Research & Development, Patton Hall P.O. Box 411 4900 Meridan St. Normal, AL 35762 -			8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211			10. SPONSOR/MONITOR'S ACRONYM(S) ARO		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S) 49366-MA-H.1		
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited					
13. SUPPLEMENTARY NOTES The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.					
14. ABSTRACT This report documents the theoretical and computational investigation of statistical pattern recognition techniques and Bayesian Networks (BN). The application of statistical pattern recognition methodology and Bayesian Networks to automatic fault diagnostics, fault prognostics, and condition based maintenance (CBM) is explored. The theory of Margin-Setting, a new pattern recognition method developed by researchers at Alabama A&M University, is documented and its applicability to problems of interest to Army is investigated. Extensive parametric studies of Margin-Setting have been carried out through training and testing the algorithms with real and simulated data. Effects of various parameters including size of the training					
15. SUBJECT TERMS Pattern Recognition, Supervised Learning, Bayesian Networks, Condition Based Maintenance					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Kaveh Heidary
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER 256-372-5587

Report Title

Mathematical Frameworks for Diagnostics, Prognostics and
Condition Based Maintenance Problems

(W911NF-05-1-0426)

Final Report
August 15, 2008

Prepared for:
Dr. David Chris Arney
US Army Research Office
PO Box 12211
Research Triangle Park, NC 27709-2211
Phone: 919-549-4254
Email: david.arney1@us.army.mil

ABSTRACT

This report documents the theoretical and computational investigation of statistical pattern recognition techniques and Bayesian Networks (BN). The application of statistical pattern recognition methodology and Bayesian Networks to automatic fault diagnostics, fault prognostics, and condition based maintenance (CBM) is explored. The theory of Margin-Setting, a new pattern recognition method developed by researchers at Alabama A&M University, is documented and its applicability to problems of interest to Army is investigated. Extensive parametric studies of Margin-Setting have been carried out through training and testing the algorithms with real and simulated data. Effects of various parameters including size of the training set, the evolution of prototypes, threshold level, margin value, etc. on the accuracy of the algorithm will be studied for various types of classification problems. The interplay between false positives and false negatives as they relate to system parameters and the application environment will be studied. Algorithms for mapping the data sets of large-scale Bayesian Network graph structures in a parallel and distributed computing environment were researched. In support of the Condition Based Maintenance (CBM) philosophy, a theoretical framework and algorithmic methodology for obtaining useful diagnostic and prognostic data from electro-mechanical systems was developed. The methods are based on vibration and modal analyses of the physical components. To illustrate the concept of the derived process, two "real world" models, a PCI circuit card, and an example rotor hub were considered. Models were created using finite element analysis (FEA) techniques, and analyzed to determine fundamental mode shapes and vibration frequencies. The results yielded vibration modes characteristic of both undamaged and damaged systems. An $O(N \log N)$ pattern recognition technique was utilized for signal discrimination. Application of the developed techniques proved very useful, and correctly identified all inserted FAULTS on the simulated systems. PIs have involved a number of advanced undergraduate students in the Department of Electrical Engineering at Alabama A&M University throughout this project. Some electrical engineering seniors participated in this project through the required Senior Design Project which is a two-semester course sequence. A course in pattern recognition was developed for senior level and beginning graduate students.

List of papers submitted or published that acknowledge ARO support during this reporting period. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Number of Papers published in peer-reviewed journals: 0.00

(b) Papers published in non-peer-reviewed journals or in conference proceedings (N/A for none)

Number of Papers published in non peer-reviewed journals: 0.00

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts): 0

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts): 0

(d) Manuscripts

Number of Manuscripts: 3.00

Number of Inventions:

Graduate Students

<u>NAME</u>	<u>PERCENT SUPPORTED</u>
FTE Equivalent:	
Total Number:	

Names of Post Doctorates

<u>NAME</u>	<u>PERCENT SUPPORTED</u>
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

<u>NAME</u>	<u>PERCENT SUPPORTED</u>
FTE Equivalent:	
Total Number:	

Names of Under Graduate students supported

<u>NAME</u>	<u>PERCENT SUPPORTED</u>
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period: 0.00

The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:..... 0.00

Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale): 0.00

Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense 0.00

The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields: 0.00

Names of Personnel receiving masters degrees

NAME

Total Number:

Names of personnel receiving PhDs

NAME

Total Number:

Names of other research staff

NAME

PERCENT SUPPORTED

FTE Equivalent:

Total Number:

Sub Contractors (DD882)

Inventions (DD882)

Mathematical Frameworks for Diagnostics, Prognostics and
Condition Based Maintenance Problems

(W911NF-05-1-0426)

Final Report
August 15, 2008

Prepared for:
Dr. David Chris Arney
US Army Research Office
PO Box 12211
Research Triangle Park, NC 27709-2211
Phone: 919-549-4254
Email: david.arney1@us.army.mil

by:
Kaveh Heidary, Professor
Andrew Scott, Asst. Prof.
Department of Electrical Engineering
Alabama A&M University
Rm. 209, Engineering and Technology Building (ETB)
Normal, AL 35762
Phone: 256-372-5587
Email: kaveh.heidary@aamu.edu

Abstract

This report documents the theoretical and computational investigation of statistical pattern recognition techniques and Bayesian Networks (BN). The application of statistical pattern recognition methodology and Bayesian Networks to automatic fault diagnostics, fault prognostics, and condition based maintenance (CBM) is explored. The theory of Margin-Setting, a new pattern recognition method developed by researchers at Alabama A&M University, is documented and its applicability to problems of interest to Army is investigated. Extensive parametric studies of Margin-Setting have been carried out through training and testing the algorithms with real and simulated data. Effects of various parameters including size of the training set, the evolution of prototypes, threshold level, margin value, etc. on the accuracy of the algorithm will be studied for various types of classification problems. The interplay between false positives and false negatives as they relate to system parameters and the application environment will be studied. Algorithms for mapping the data sets of large-scale Bayesian Network graph structures in a parallel and distributed computing environment were researched. In support of the Condition Based Maintenance (CBM) philosophy, a theoretical framework and algorithmic methodology for obtaining useful diagnostic and prognostic data from electro-mechanical systems was developed. The methods are based on vibration and modal analyses of the physical components. To illustrate the concept of the derived process, two “real world” models, a PCI circuit card, and an example rotor hub were considered. Models were created using finite element analysis (FEA) techniques, and analyzed to determine fundamental mode shapes and vibration frequencies. The results yielded vibration modes characteristic of both undamaged and damaged systems. An $O(N \cdot \log N)$ pattern recognition technique was utilized for signal discrimination. Application of the developed techniques proved very useful, and correctly identified all inserted FAULTS on the simulated systems. PIs have involved a number of advanced undergraduate students in the Department of Electrical Engineering at Alabama A&M University throughout this project. Some electrical engineering seniors participated in this project through the required Senior Design Project which is a two-semester course sequence. A course in pattern recognition was developed for senior level and beginning graduate students.

Table of Contents

Abstract	2
Table of Contents	3
Introduction	5
Margin-Setting with Hyperellipsoidal Surfaces	7
Introduction	7
New Approach.....	8
Problem Formulation.....	8
Testing the Ellipsoidal Approach to Margin Setting	13
Conclusions	34
Application of Fuzzy Clustering to Color Image Segmentation.....	35
Mathematical Formulation	47
Simulation Results.....	50
Bayesian Network Modeling	61
Introduction	61
Bayesian Networks and Decision Graphs for Complex Systems	63
Modeling Framework	66
BN – Graph Partitioning for Simulation in a Distributed Environment	74
BN Graph Decomposition Algorithms	76
Graph Representation of BN Model.....	77
Partitioning Algorithm Analysis.....	83
BN_Graph Reintegration.....	86
Reintegration Algorithm Development	86
Reintegration Algorithm Analysis.....	89
Algorithm Application.....	91
Conclusions of BN-Graph Algorithms	95
Electro-Mechanical Diagnostics/Prognostics	97
Introduction	97
Background	98
Problem Definition	101
Analysis.....	101
Modal Analysis.....	101
Sensor Selection and Placement	108
Signal Processing.....	108
Results	115
PCI Card Simulation.....	115

Rotor Hub Simulation.....	126
Pattern Recognition for Dynamic Cracking.....	135
Conclusions	137
Student Project 1: SHELFP – Data Mining of ATS and UUT Information.....	140
Background	140
Approach:	141
System Overview:	143
Design Considerations:.....	144
Database Structure.....	145
The Relational Database System in SHELFP.....	149
System Architecture	152
Conclusions	156
Student Project 2 – Intelligent Sensors.....	158
Background	158
Embedded Fatigue Sensor	158
Sensor Selection and Placement	160
Fatigue Counter Development.....	163
Active RFID	167
Intelligent Sensor Results	169
Intelligent Sensor Conclusions	170
Pattern Recognition Course.....	171
Course Description	172
Graduate Program Description	174
Conclusions and Future Work	179
Bibliography.....	182

Introduction

Research efforts supported by this award have involved two faculty members from the Department of Electrical Engineering at Alabama A&M University. The thrust of constituent research projects has been directed towards development, formulation, and implementation of new algorithms with potential applications to automatic fault diagnostics, prognostics, and condition based maintenance. Each year two or more three-member teams of electrical engineering students have participated in research and development activities derived from this effort as part of the required two-semester Senior Design Project course sequence. Research and development activities associated with this award have resulted in advancing state of knowledge in areas of pattern recognition, supervised learning, self organizing clusters, and Bayesian networks. Student participations in research projects associated with this effort have led to senior design projects in areas of importance to industry and government. This has helped the EE Department to equip its BSEE graduates with skill sets that make them better prepared for industrial and government R&D careers as well as pursuit of advanced degrees.

The work in the area of pattern recognition and supervised learning has resulted in significant improvements in the Margin Setting algorithms developed previously at AAMU. The new approach utilizes ellipsoidal surfaces in the feature space as class separators. This approach results in production of more robust classifiers with smaller number of classifier filter rounds compared to the original formulation. Examples using simulated and real data show the superiority of this approach. It has been shown that the performance of the ellipsoidal classifier approaches that of the optimal classifier. This has been demonstrated with multi class problems using training data obtained from normal distributions with known statistics, where the training algorithm is oblivious to the actual statistical distributions of the training data. Applications of this algorithm include fault diagnostics and prognostics.

The work in the area self organizing clusters involves application of Fuzzy Clustering to unsupervised color image segmentation. Given a set of data points in multidimensional feature space, and a user-specified integer representing number of classes, the algorithm computes a set of prototypes and a membership matrix. Each prototype is a vector in the feature space and is the optimal representation of the corresponding class. Each element of the membership matrix represents the degree of membership (association) of a data points in the respective cluster. In the examples presented here, the algorithm is used for unsupervised image segmentation based on pixel RGB coordinated. This algorithm has applications in automatic fault diagnostics where the training data sets are not labeled with respect to class disposition.

The work in the area of Bayesian networks (BN) reviews previous efforts and investigates additional requirements necessary to construct viable models of military system(s) and/or operations, and apply Bayesian methodologies to those models to obtain

pertinent information regarding potential failure and performance as a function of the current state of the system(s), and relevant archived experiential data of the system(s). Representation of the networks as directed acyclic graph structures, and techniques for decomposing the systems are examined.

In support of the efforts for Condition Based Maintenance (CBM) based on prognostics and diagnostics, this project examines techniques for non-invasive assessment of the physical condition of system components. Techniques for sensing component fatigue and vibration modes are discussed. Effects on the component modal characteristics due to damage/deterioration from environmental and external forces are examined. A process, with low computational complexity, for detection of anomalous modal behavior due to degraded physical condition is presented.

Additional efforts in CBM are documented. Two projects in which students were heavily involved are discussed. First, development of a software tool called S-HELP (System Health by Enabling Learning and Prognostics) is detailed. The software utilizes Bayesian methodologies can be used for adaptive learning in embedded diagnostic systems. In this framework, the overall aim is to develop probabilistic models that are well matched to the data, and make optimal predictions with those models. The second project involving a conceptual electronic monitoring system based on the integration of small accelerometer sensors, field programmable gate arrays (FPGAs) and wireless radio frequency identification (RFID) technologies is discussed.

Finally, a course in Pattern Recognition was developed within the Department of Electrical Engineering. The course is designed for advanced senior students, and graduate students. It is presently classified as a “Special Topics” course, EE 490 for undergraduate students, and EE 590 for graduate students. It is anticipated that the pattern recognition course will play a strong role in the EE core component for educating students in our new Masters program. A detailed outline of the Pattern Recognition course is presented, followed by a description of the graduate program.

Margin-Setting with Hyperellipsoidal Surfaces

In a number of prior papers we described and explored a new pattern recognition method called Margin-Setting that accomplishes excellent generalization using very few training samples. The result was a multi-round classifier with each round consisting of a set of hyperspheres such that if a datum fell within a certain hypersphere, it was labeled with one particular class. Margin-Setting achieves concurrent low Vapnik-Chervonenkis (VC) dimension and high accuracy which is a consequence of partitioning the training set into smaller sets that make this possible. This paper extends Margin-Setting from hyperspheres to hyperellipsoids resulting in improved performance.

Introduction

Machine learning algorithms are processes that generate functions based on pairs of vectors representing the input objects and corresponding desired outputs. In regression analysis the function outputs are continuous, and in classification they are class labels. The goal of statistical pattern recognition and supervised learning is to place new, previously-unseen objects into their proper classes with high reliability. This is done by generalizing from what can be learned using a set of known and labeled training elements (trainers), each associated with one of two or more classes (Schuermann, 1996) and (Duda et.al., 2001). Each trainer can be viewed a point in the multidimensional feature hyperspace. Support Vector Machines (SVM), are maximum margin classifiers and are widely used in many applications (Taylor and Cristianini, 2000), (Vapnik, 1995 and 1998) and (Burgess, 1998). Strength of SVM-based classifiers has been empirically established on many benchmark classification problems including handwritten character recognition (Tapiador, et.al., 2004).

This paper presents an algorithm for development of a powerful multi-class classifier that combines elements of both SVM and Boosting (Tapiador, et.al., 2004). The classifier presented here is a natural extension and a generalization of an earlier classifier (Schapire, 1990) that outperforms SVM on the widely used Fisher iris data set (Valiant, 1984). Margin-Setting has been used to train spatial (Valiant, 1984) and spectral (Caulfield and Heidary, 2005) classifiers capable of solving challenging classification problems. One of the useful properties of Margin-Setting is the flexibility with which one can tradeoff misclassification and non-classification error rates.

Margin-Setting partitions the training set in a hierarchical order such that exemplars can be perfectly separated by user-prescribed decision hyper-surfaces and margins. It accomplishes this by leaving out hard-to-classify data from the training set and keeping only those trainers that can be classified with the prescribed surface and margin, in the first round. Trainers classified by round-one classifier are removed from the training set and the round-two classifier is trained with the remaining trainers employing the same process, whereby hard to classify trainers are removed to be utilized in the next training

round. The process continues until the training set is empty, resulting in a multi-round classifier. Margin-Setting as described in (Schapire, 1990), results in hyperspherical separating surfaces. The classifier consists of multi-round composite filters. A composite filter is comprised of prototype-radius pairs associated with each class whose trainers are present during that training round. This paper develops an algorithm for training a multi-round classifier whose composite filters are sets of hyperellipsoids.

New Approach

The multi-round classifier described here consists of an ordered set of composite filters. Each composite filter comprises a set of hyperellipsoidal decision surfaces. Henceforth hyperellipsoid and ellipsoid will be used interchangeably. We define a hyperellipsoid in N-dimensional hyperspace as the locus of points such that the sum of distances from any point on that surface to each of up to N points called foci is a constant we call the radius. Each set of foci thus determines a distance measure: i.e. the sum of Euclidean distances from the foci, with respect to points in the feature space. A point in feature space whose distance with respect to a set of foci is less than the radius is said to fall within the volume of the respective ellipsoid. Each ellipsoid is associated with a particular class. Ellipsoidal volumes corresponding to the same composite filter (filter round) are not necessarily mutually exclusive and may have non-zero intersections.

A previously unencountered object (untrained-on vector in feature space) is tested with the round-one composite filter. If it falls within the volume of any one of the round-one ellipsoids using the distance measure just described, it is labeled with the corresponding class and the classification process is terminated. If not, it is passed to the next round and so on until the last round. If the object fails to be captured by any ellipsoid it remains unclassified. In any particular round an object may fall within multiple ellipsoids representing different classes. In such cases, the object is assigned to the class with respect to which it has the largest proximity factor which is defined in Section 3.

The training process starts with the initial finite trainer set and obtains one ellipsoid for each class. An ellipsoid is specified by its foci (points in feature space) and its radius. The set of ellipsoids computed with the entire trainer set (complete set of exemplars) constitutes the round-one composite filter (classifier). Trainers that are contained within the volumes of ellipsoids constituting the round-one filter are removed from the training set. The above process is repeated using the reduced training set. A set of ellipsoids constituting the round-two filter is computed. The process is continued until the training set is empty or it contains trainers of a single class. If the trainer set consists of trainers from a single class, the minimum-radius ellipsoid containing all remaining trainers constitutes the last-round classifier.

Problem Formulation

A set of vectors in the multidimensional feature space, each labeled with a certain class brand, constitutes the training set. The objective is to develop a system capable of placing unlabeled vectors in their proper classes with high reliability, based on what can be

learned from the set of labeled exemplars. The resulting system is a multi-round classifier, with each classifier-round consisting of a composite filter comprised of a set of hyperellipsoids. A hyperellipsoid is specified by a set of N_f points in N_d -space, called foci, and a radius. It is the locus of points in the feature space such that the sum of their distances to the foci is equal to the radius. Number of foci in this paper is set to two ($N_f=2$), and Euclidean distance is used as the distance measure.

Each round of training begins with a set of trainers that have not been classified (captured) by filters associated with previous rounds. This leads to a composite filter and a reduced set of exemplars to be passed on to the next training round. The composite filter in each round consists of multiple ellipsoids, one associated with each class represented by current trainers. Computation of a typical ellipsoid is equivalent to ascertaining the respective foci-pair and radius using the procedure described below. In order to illustrate the training process certain definitions and terminology are explained first.

We define point-ellipsoid distance as the sum of Euclidean distances between the point in the feature space and the ellipsoid foci.

$$D_p = \sqrt{\sum_{i=1}^{N_d} (p_i - f_{1,i})^2} + \sqrt{\sum_{i=1}^{N_d} (p_i - f_{2,i})^2} \quad \text{Eq. 1}$$

Where p , f_1 , f_2 , N_d denote, respectively, arbitrary point, foci-pair, dimensionality of the feature space, and p_i , $f_{1,i}$, $f_{2,i}$ are coordinates of respective points. It is noted that point-ellipsoid distance as defined here is foci-dependent only and is independent of ellipsoid radius. If the point-ellipsoid distance is greater (smaller) than the ellipsoid radius, the point is said to be outside (inside) the ellipsoid. A class-ellipsoid denotes any ellipsoid whose foci are derived from trainers of that particular class, and does not contain within its volume any trainers from other classes. The zero-margin radius of a typical class-ellipsoid is the minimum distance between that ellipsoid and all trainers associated with other classes.

$$R_{0,m}^{(j)} = \min_{p \in (T \setminus T^{(j)})} (D_{m,p}^{(j)}) \quad \text{Eq. 2}$$

where T , $T^{(i)}$ denote the set of all current trainers and class- j trainers, respectively, $D_{m,p}^{(j)}$ is the distance between trainer- p and ellipsoid- m of class- j , and $R_{0,m}^{(j)}$ is zero-margin radius of the said ellipsoid. Figure 1 illustrates the concept of trainer-ellipsoid distance and zero-margin radius for a 2D feature space, where class ellipsoids are ellipses. In this figure trainers associated with classes one and two are represented with circles and squares, and class ellipses are shown with solid and dashed lines, respectively. Trainer-ellipsoid distance here is the sum of distances between the trainer and the foci-pair, and zero-margin radius of a typical ellipse is the minimum trainer-ellipse distance with respect to all other-class trainers. It is noted that the foci do not necessarily coincide with the actual trainers.

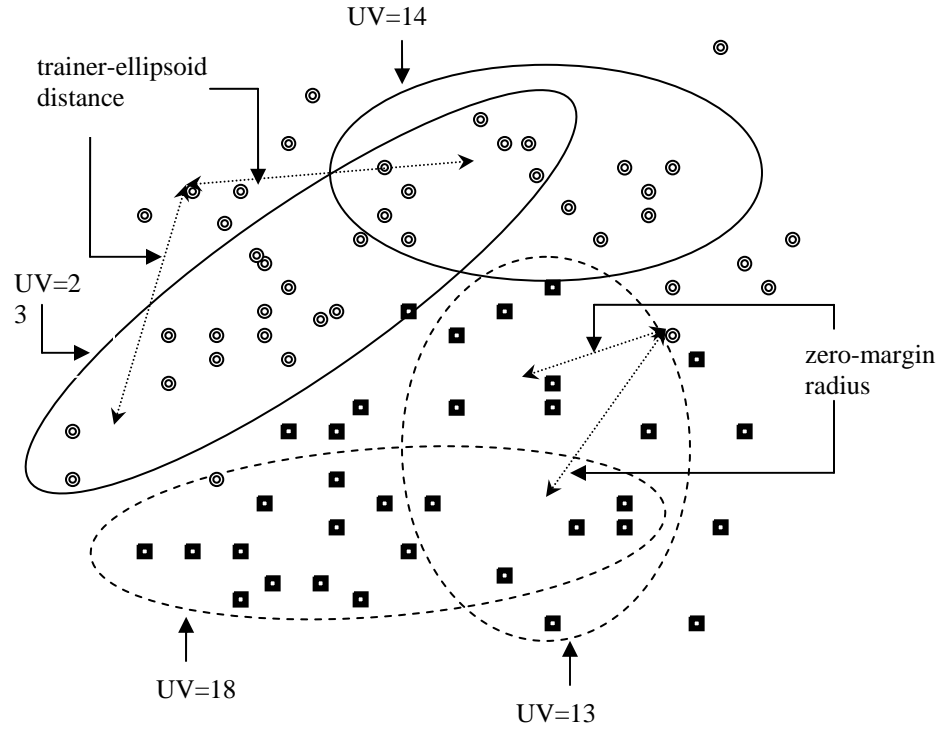


Figure 1: Exemplars of a two-class problem in R2 feature space, represented as squares and circles, with two typical zero-margin ellipses and respective utility-values for each ellipse

Foci of a class-ellipsoid are obtained from mutations of the class trainers as described later. Ellipsoid radius is always less than or equal to its zero-margin radius. The user-specified *percent-margin* parameter determines the margin by which a class ellipsoid avoids containing other-class trainers within its volume, by commensurately reducing the zero-margin radius. The utility value of an ellipsoid is the number of same-class trainers (same class as the ellipsoid) that are contained within the ellipsoid. Utility value of a typical ellipsoid is computed using its zero-margin radius. A trainer is contained within an ellipsoid if the trainer-ellipsoid distance as defined in (1) is less than the ellipsoid radius.

$$u_m^{(j)} = |S_m^{(j)}|; \quad S_m^{(j)} \subseteq T^{(j)}; \quad S_m^{(j)} = \left\{ p^{(j)} \left| D_{m,p^{(j)}}^{(j)} < R_{0,m}^{(j)}, \quad p^{(j)} \in T^{(j)} \right. \right\} \quad \text{Eq. 3}$$

Where $u_m^{(j)}$ is the utility value of ellipsoid-m of class-j, $|x|$ denotes cardinality of set x , $p^{(j)}$ is a class-j trainer, $D_{m,p^{(j)}}^{(j)}$ is the trainer-ellipsoid distance, and the rest of parameters are defined as before. A class ellipsoid is computed by first generating an assortment of

ellipsoids each based on a foci-pair obtained directly from the remaining class trainers. Each of these class ellipsoids consists of one foci-pair and the corresponding zero-margin radius. Typical foci-pairs are obtained by random pairing of class trainers. Utility value of class ellipsoid is computed as put forth in (3). The full set of foci-pairs for a particular class consists of all possible two-trainer combinations of current class trainers. For large training sets using all possible trainer combinations may lead to prohibitively large sets of ellipsoids. When number of trainers is excessively large a user-specified number (i.e. 1000) of class ellipsoids are generated by random pairing of trainers.

$$F_0^{(j)} = \left\{ (p_k^{(j)}, p_l^{(j)}) \mid 1 \leq k, l \leq M^{(j)}, M^{(j)} = |T^{(j)}|, p_i^{(j)} \in T^{(j)} \right\} \quad \text{Eq. 4}$$

Where $p_k^{(j)}$ denotes trainer-k of class-j, and $F_0^{(j)}$ represents the set of all generated class-j foci-pairs in the current round. It is noted that both constituents of some foci-pairs may correspond to the same trainer, in which case the ellipsoid degenerates into a sphere. The cardinality (population) of the full set of foci-pairs is

$$|F_0^{(j)}| = \frac{M^{(j)}(M^{(j)} + 1)}{2} \quad \text{Eq. 5}$$

Where $M^{(j)}$ denotes number of class-j trainers. If the full-set foci-pair population is greater than the user-specified parameter $N_e^{(0)}$, a number of foci-pairs (equal to $N_e^{(0)}$) are randomly selected from the full set in (4). The zero-order class-ellipsoid is synthesized from the foci-pair with the highest utility. If there are multiple pairs with such property one of them is chosen randomly. The class-ellipsoid is uniquely specified by its pair of foci and its zero-margin radius, which was defined previously. It is noted that in problems with very large number of trainers, where examination of the complete set of foci-pairs is computationally prohibitive, the set of foci-pairs based on random pairing of trainers leads to near-optimal ellipsoids. This is achieved due to the fact that although the foci-pairs are initially chosen from the trainer set, they are evolved through the mutation process described below. While the initial set of ellipsoids may not include the one with the highest possible utility value, the mutation process adjusts the foci in such a manner that utility tends to increase towards its maximum possible value.

In order to optimize class-ellipsoids an evolutionary process analogous to animal immunological system development is used (Cowell et.al., 1999) to obtain class-ellipsoids and classifier filters with superior characteristics. Class ellipsoids with desired characteristics (higher utility values) are regenerated (mutated) preferentially leading to overall improvements in their performance. Preferential treatment of class ellipsoids with desirable traits, by way of spawning larger number of progenies, is akin to development of immunological systems (Cowell et.al., 1999). The set of foci-pairs is mutated into a larger set of new foci-pairs by mutation of each pair into a number, commensurate with utility value of the respective ellipsoid. Subsequent to mutation, $N_e^{(0)}$ foci-pairs are randomly selected from the large set of mutated foci-pairs spawned from the zero-order

foci-pairs in order to form the first-order foci-pairs and respective ellipsoids. The first-order class-ellipsoid is synthesized from the foci-pair with the highest utility value. The mutation process continues for a user-specified number of transmutations or until maximum utility value reaches a plateau. Following the last mutation order, the foci-pair with highest utility value is identified and the class-ellipsoid is subsequently synthesized. Mutated foci-pairs are obtained by forming two multivariate normal distributions in the feature space, each with mean vector equal to the respective focal point, and diagonal covariance matrix. Mutated focal points are chosen randomly from the following probability distribution function.

$$f(\hat{p}) = \left(\frac{1}{2\pi} \right)^{\frac{N}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (p - \hat{p})^T \Sigma^{-1} (p - \hat{p}) \right\} \quad \text{Eq. 6a}$$

$$\Sigma = \sigma^2 I_N \quad \text{Eq. 6b}$$

$$p = [p_1, \dots, p_{N_d}] ; \quad \hat{p} = [\hat{p}_1, \dots, \hat{p}_{N_d}] \quad \text{Eq. 6c}$$

Where Σ is the covariance matrix, I_{N_d} is the identity matrix, and p , \hat{p} denote, respectively, a point and its mutation in the feature space. It is assumed that coordinates of mutated points are obtained from mutually independent normal distributions with identical user-specified variances. Each one of the two foci-pair affiliates is mutated into $N_e^{u_m^{(j)}}$ points in accordance to the normal distribution of (6a), where $u_m^{(j)}$ denotes the focal-pair utility value of (3). $N_e^{u_m^{(j)}}$ mutated foci-pairs are formed by pairing points randomly chosen from respective distribution functions. Mutated points are paired on an exclusive basis, that is, each point from the first distribution is uniquely paired with one point from the second distribution. During each training round the mutation and selection procedure just described is performed for each class with extant trainers. This process results in formation of one ellipsoid per class and culminates in construction of the composite filter in each round of training.

As a consequence of mutations each class-ellipsoid is optimized with respect to its utility value or number of class-trainers contained within its volume. The procedure described above implements the round-one training process. The round-one filter comprises one ellipsoid for each class. Each ellipsoid is represented by its foci-pair and radius $F_{(1)}^{(j)}, \hat{R}_{(1)}^{(j)}$, where superscript j refers to class index and subscript 1 denotes the round number. $F_{(1)}^{(j)}$ is a doublet in the feature space obtained from mutated trainers of the respective class, and radius is obtained by scaling the zero-margin radius using the user-specified percent-margin parameter.

$$\hat{R}_{(1)}^{(j)} = (1 - 0.01\alpha) \bar{R}^{(j)} \quad \text{Eq. 7}$$

Where $\bar{R}^{(j)}$ denotes class- j zero-margin radius of the optimized ellipsoid as defined by (2) and α is the percent-margin. Trainers contained within the volume of round-one filter are defined as follows.

$$\bar{T}_{(1)}^{(j)} = \left\{ P_i^{(j)} \mid \|P_i^{(j)} - F_{(1)}^{(j)}\| < \hat{R}_{(1)}^{(j)} \right\} \quad \text{Eq. 8}$$

The trainers captured by the round-one classifier (composite filter) as given above, are removed from the training set to form the training set for computation of the round-two filter.

$$T_{(2)}^{(j)} = T^{(j)} \setminus \bar{T}_{(1)}^{(j)} \quad \text{Eq. 9}$$

Repeating the entire training process described above with $T^{(j)}$ replaced with $T_{(2)}^{(j)}$, leads to the round-two filter. Training rounds continue, each round resulting in computation of a composite filter and a reduced training set to be passed to the next round. The process is terminated when training set is empty. This concludes the training process. The process results in a multi-round classifier of composite filters, each comprised of multiple class-ellipsoids.

Classification of an unlabeled input vector in the feature space begins with testing it with the round-one composite filter. If the distance between the input and one of the round-one ellipsoids is less than or equal to its radius, the input is labeled as member of the corresponding class and the process terminates. If not, it is passed on to the next round and the process is repeated. If the input reaches the last round and fails to be captured by any one of the ellipsoids, it remains unlabeled. An input may be captured by multiple ellipsoid of a composite filter. In that case it is assigned to the class with greatest proximity factor. Vector-ellipse proximity-factor is defined as

$$PF_i^{(j)} = 1 - \frac{\|P_i - F^{(j)}\|}{\hat{R}^{(j)}} \quad \text{Eq. 10}$$

Where $F^{(j)}, \hat{R}^{(j)}$ denote class-j foci and radius, respectively, and P_i represents trainer-i. Proximity-factor for points on the ellipsoidal surface, inside ellipsoidal volume, and outside the volume is zero, positive and negative, respectively. If a vector falls within volumes of multiple ellipsoids of a particular composite filter, it is labeled with the class that results in the highest proximity factor. In labeling input vectors a classifier may commit two types of error; namely, the input may be incorrectly labeled (misclassification error), or it may remain unlabeled (non-classification or indecision error).

Testing the Ellipsoidal Approach to Margin Setting

The examples below demonstrate effectiveness of this algorithm in solving difficult nonlinearly separable multi-class classification problems. In order to present graphical illustrations of the classifier, here we consider examples in two-dimensional (2D) feature space. Training and test elements are given as points in the xy-plane of the Cartesian coordinate system. The algorithm, however, is general and makes no distinction between two and multi-dimensional problems. In each of the following examples the training set is finite and a classifier with finite rounds, therefore, captures all the trainers.

In the first example the classifier is trained using fifty training elements from each of four classes. In Figure 2a classes one through four trainers are shown as circles, dots, triangles, and asterisks, respectively. Class-one through four trainers in this example were obtained from four random distributions centered at (0,0), (1,0), (0,1), and (1,1), respectively. Each distribution is a mutually independent 2D Gaussian function, with standard deviations along x, y coordinates for classes one through four equal to (0.3,0.1), (0.2,0.25), (0.1,0.25), and (0.25,0.25). Figures 2b-d show the multi-round classifier trained with twenty-five-percent margin-value. It is noted that each classifier-round, in general, is a composite filter comprised of multiple hyperellipsoids in the feature space, which for the 2D case are represented as ellipses in the xy-plane. Parameters of ellipses comprising the composite filters of the multi-round classifier are also tabulated. Each row of Table 1 corresponds to data for an ellipse which forms a component of a single-round composite filter. First row, for example, refers to the round-one class-one filter and tabulates the xy-coordinates of two focal points, (F1-x,F1-y) and (F2-x,F2-y), magnitude of the radius, A, and number of class-one trainers it embraces, N. A0 and N0 represent, respectively, radius and number of trainers corresponding to the zero-margin filter. The round-one composite filter is trained using the entire trainer set of fifty elements from each class. Figure 2b and Table 1 show that the round-one composite filter subsumes a large portion of the trainers, namely, 48, 45, 47, and 44 from classes one through four, respectively. The round-two composite filter, on the other hand, is trained on the remaining trainers, namely 2, 5, 3, and 6 trainers from classes one through four, respectively. It is seen that all class-one and class-three trainers are subsumed by the first two composite filters. The round-three filter, therefore, is trained exclusively with elements from classes two and four.

Table 1: Classifier parameters for the example of Figure one

round	F1-x	F1-y	F2-x	F2-y	A0	A	N0	N	class
1	-0.5624	-0.1859	0.1740	0.0230	1.5203	1.3316	50	48	1
1	1.0760	-0.0760	1.0760	-0.0760	1.2399	0.9299	49	45	2
1	-0.0630	1.2709	-0.1792	0.7279	0.9937	0.8841	50	47	3
1	0.7414	1.0781	1.5219	1.8469	2.0164	1.7862	49	44	4
2	-0.9411	-0.1222	0.4959	0.1865	1.8003	1.7176	2	2	1
2	0.9809	-0.5459	1.1475	0.4083	1.2857	1.2064	4	3	2
2	-0.0124	0.5744	-0.0124	0.5744	0.9904	0.7428	3	3	3
2	0.7799	0.5198	1.7912	0.9160	1.1140	1.1070	5	4	4
3	1.3109	0.5950	0.9665	0.4855	0.6484	0.5767	2	2	2
3	0.2906	0.9660	0.8200	0.4904	0.9759	0.9098	2	2	4

In order to assess its performance, the above classifier was used to classify forty-thousand previously unseen elements. The set of test objects was comprised of ten-thousand elements obtained randomly from each of the probability distribution functions described previously. It is noted that the classifier training process is oblivious to the said distribution functions and was trained exclusively on the two-hundred labeled objects shown in Figure 2a.

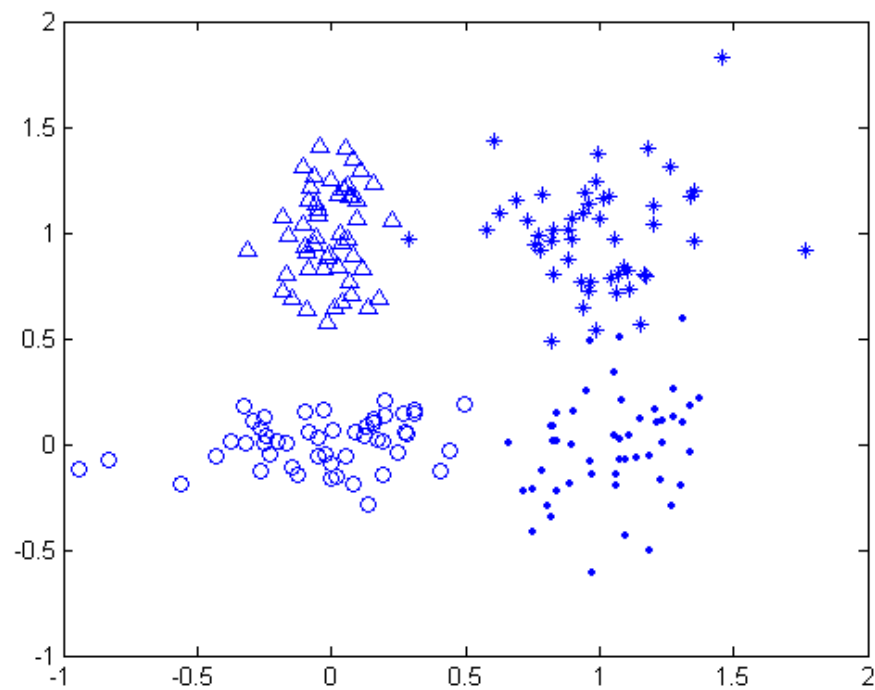


Figure 2a: Fifty labeled elements from each of four classes used to train the classifier

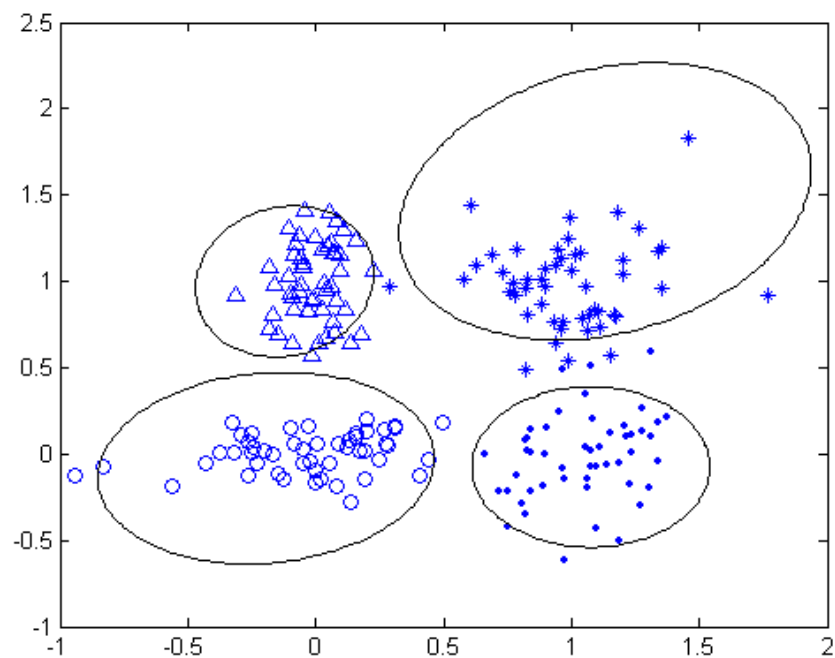


Figure 2b Round-one composite filter

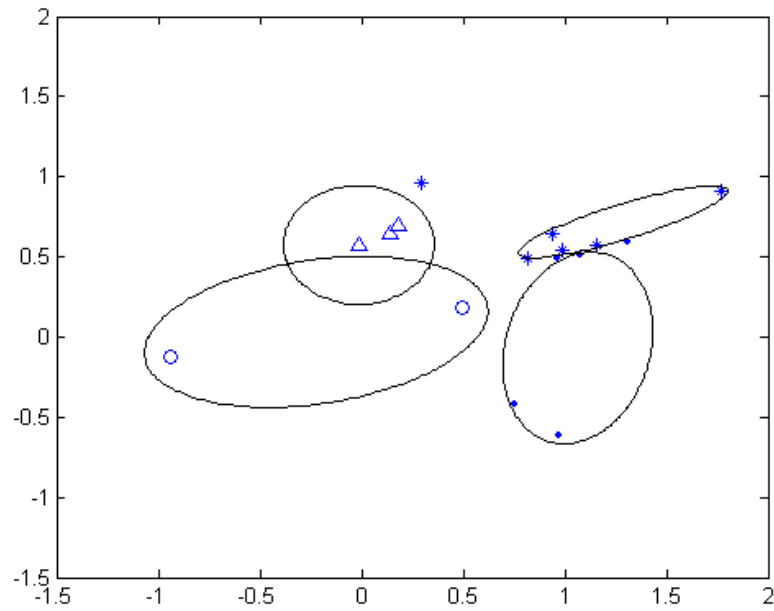


Figure 2c Round-two composite filter

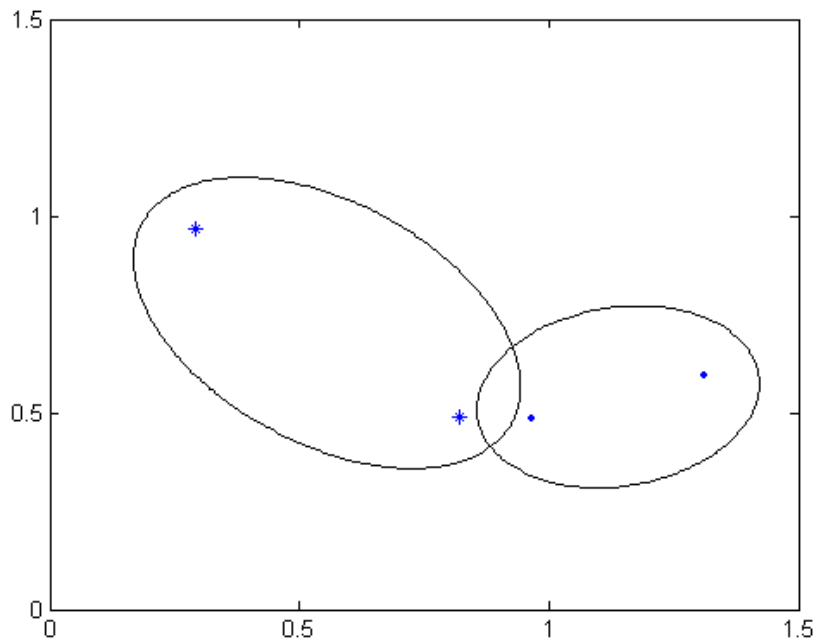


Figure 2d Round-three composite filter

Test results are given in Table 2, where numbers of test elements captured by each classifier round are shown in parentheses. It is seen from Table 2 that the classifier

performs well in that it results in concurrent low misclassification and indecision rates despite the small number of trainers. Of the forty-thousand test elements 37 421 were labeled correctly, 1 080 were misclassified and 1 499 elements remained unclassified. Of the ten-thousand class-two test elements, for example, 8 977 were correctly labeled, 128, 303 were mislabeled as class-one and class-four, respectively, and 592 remained unclassified. Of the correctly classified class-two elements 8 497 were captured by the round-one filter, 415 by round-two and 65 by round-three. Of the 303 class-two objects misclassified as class-four, the number of elements captured by round-one through three class-four filters (ellipses) are, respectively 37, 119, and 147.

Table 2: Class assignments for ten-thousand untrained-on elements from each class

	Class-one (R1/R2/R3)	Class-two (R1/R2/R3)	Class-three (R1/R2/R3)	Class-four (R1/R2/R3)	Unclassified
Class-One	9 645 (9 267/378/0)	190 (190/0/0)	0	0	165
Class-two	128 (22/106/0)	8 977 (497/415/65)	(8 0	303 (37/119/147)	592
Class-three	159 (158/1/0)	0	9 154 (8 539/615/0)	55 (2/0/53)	632
Class-four	5 (1/4/0)	233 (41/100/92)	7 (2/5/0)	9 645 (8 735/540/370)	110

In order to compare performance of margin-setting-based classifiers with hyperellipsoidal decision surfaces with those with hyperspherical decision surfaces (Caulfield and Heidary, 2005), the exemplar set of Figure 2 was used to train a twenty-five-percent margin classifier comprised of circular filters. The resulting classifier is a ten-round composite filter compared with the three-round filter described above. Table 3 lists correct classification, misclassification, and indecision rates for the two classifiers. It is seen that ellipsoidal classifiers result in superior performance. In addition to lower error rates ellipsoidal surfaces result in simpler classifiers with a smaller number of composite filters.

Table 3: Performance comparisons between elliptical and circular classifiers

	correct classification rate	misclassification rate	Indecision rate
ellipse	93.55	2.70	3.75
circle	91.07	2.89	6.04

Figure 3a-3d show the effect of trainer set population and margin value on the overall classifier performance. In these examples the classifier was trained using labeled samples from each of the four classes described earlier and was tested with ten-thousand random samples obtained from each class distribution function. For each setting of the trainer set population and margin value the simulation was repeated ten times and results were averaged in order to obtain statistically reliable data. For each simulation a newly generated trainer set was used to train the classifier. It is seen that in this example the overall correct classification rate is fairly independent of margin value. The misclassification rate, in general, decreases with margin while indecision increases as expected.

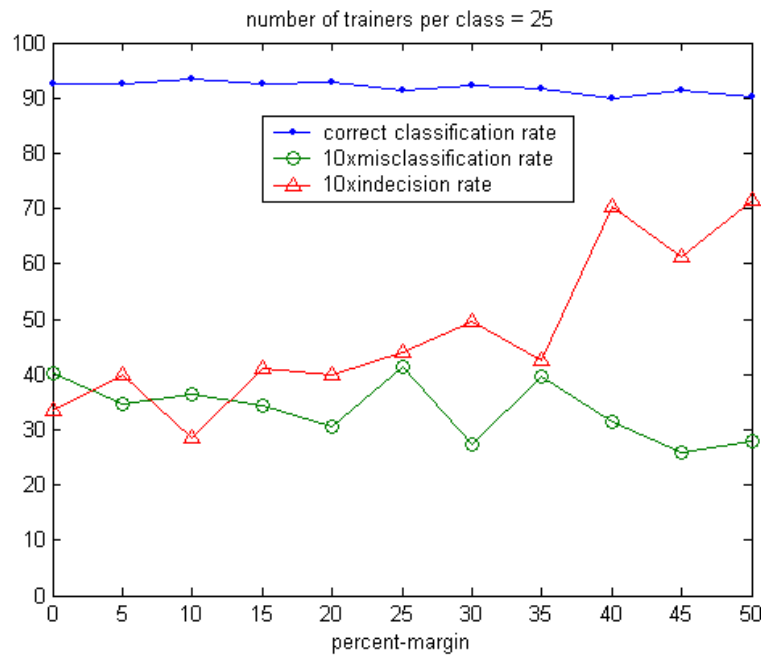


Figure 3a: Effect of margin-value on classifier performance

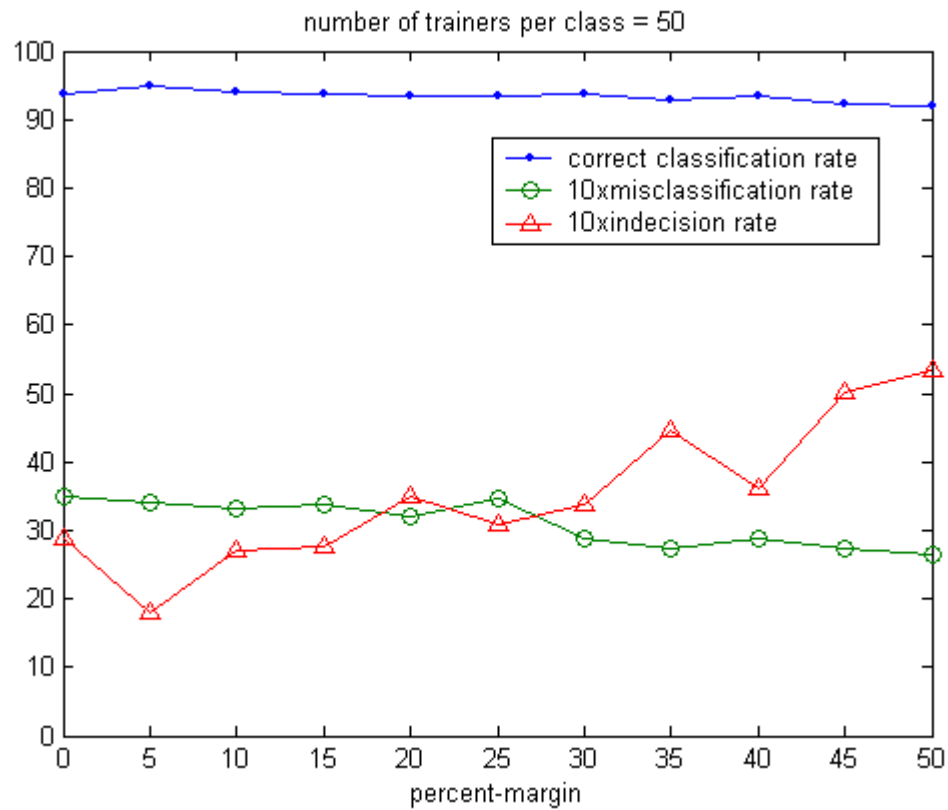


Figure 3b Effect of margin-value on classifier performance

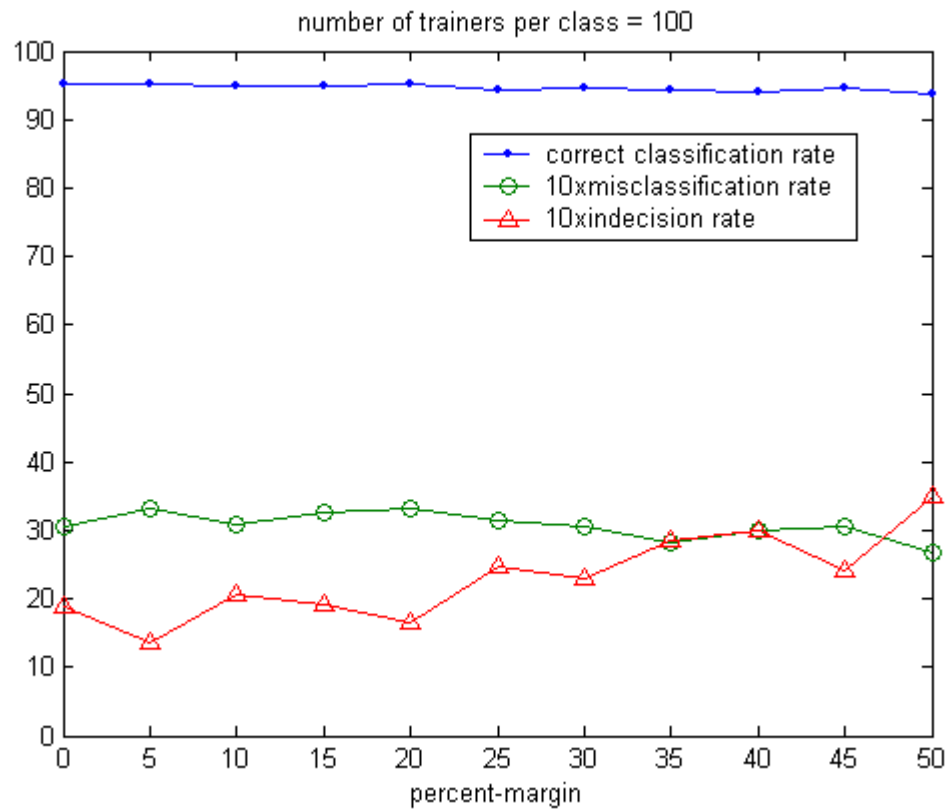


Figure 3c Effect of margin-value on classifier performance

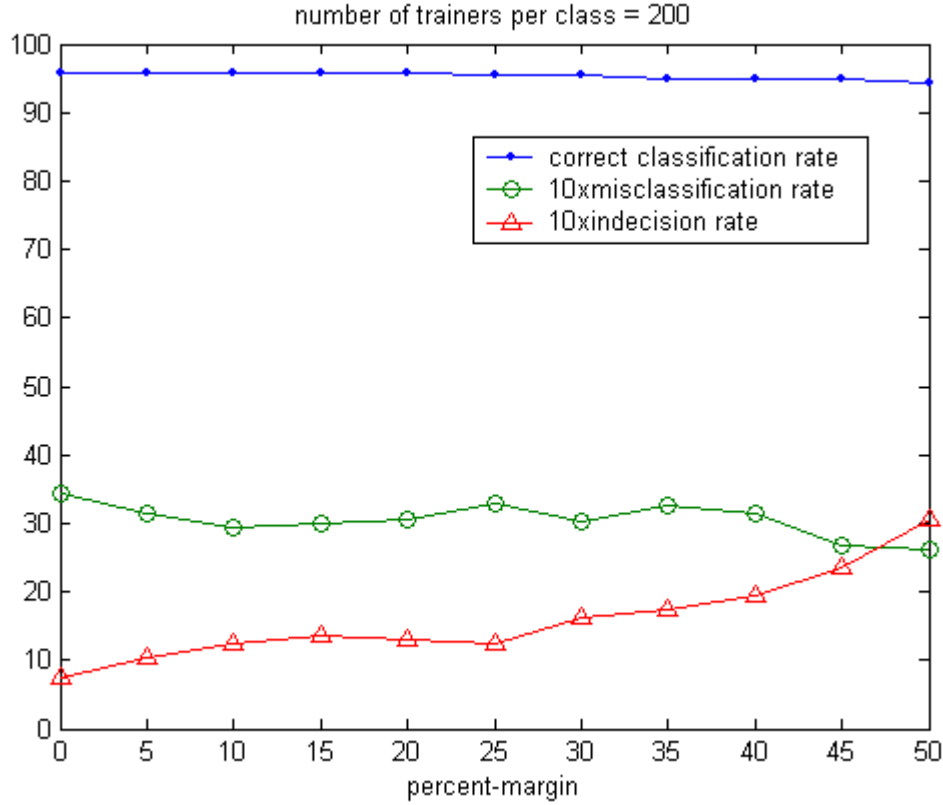


Figure 3d Effect of margin-value on classifier performance

The next example considers training and testing the classifier where each training object is labeled with one of two classes. In order to display the classifier we consider a 2D feature space similar to the previous example. Training and test objects are obtained from the following normal distributions.

$$r_1 \sim N(R_1, \sigma_{r_1}^2) \quad \text{Eq. 11a}$$

$$\phi_1 \sim N(\Phi_0, \sigma_{\phi_1}^2) \quad \text{Eq. 11b}$$

$$r_2 \sim N(R_2, \sigma_{r_2}^2) \quad \text{Eq. 11c}$$

$$\phi_2 \sim N(\Phi_0, \sigma_{\phi_2}^2) \quad \text{Eq. 11d}$$

Where subscripts 1, and 2 denote class-one and class-two, respectively, and r, ϕ are the polar coordinates of samples which are represented as points in the plane. The parameters for the example presented here are $R_1=1, R_2=1.5, \Phi_0=\pi/4, \sigma_{r_1}=0.1, \sigma_{r_2}=0.15, \sigma_{\phi_1}=\pi/16, \sigma_{\phi_2}=\pi/8$. One hundred random samples from each set were used to train the classifier with a margin value of ten-percent. The composite filters of the four-round classifier and the training samples utilized during respective training rounds are shown in Figures 4a-d. The classifier was trained utilizing one-hundred labeled samples and no other information about the probability distribution functions.

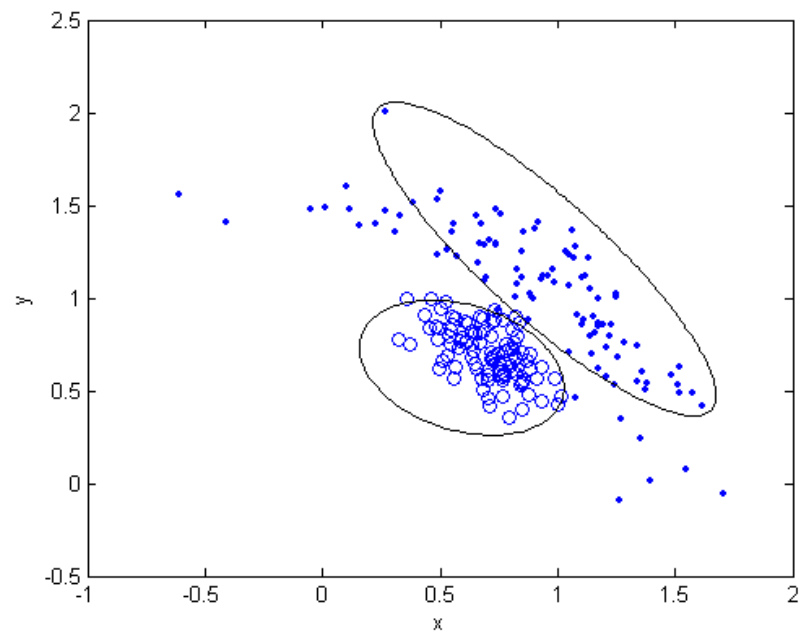


Figure 4a: Entire trainer set and round-one composite filter

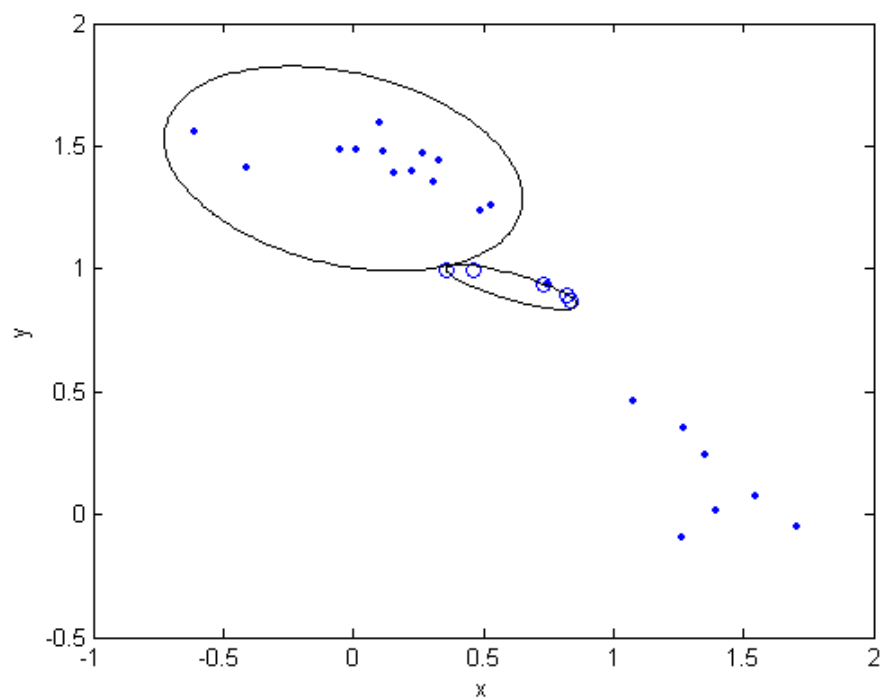


Figure 4b Round-two composite filter

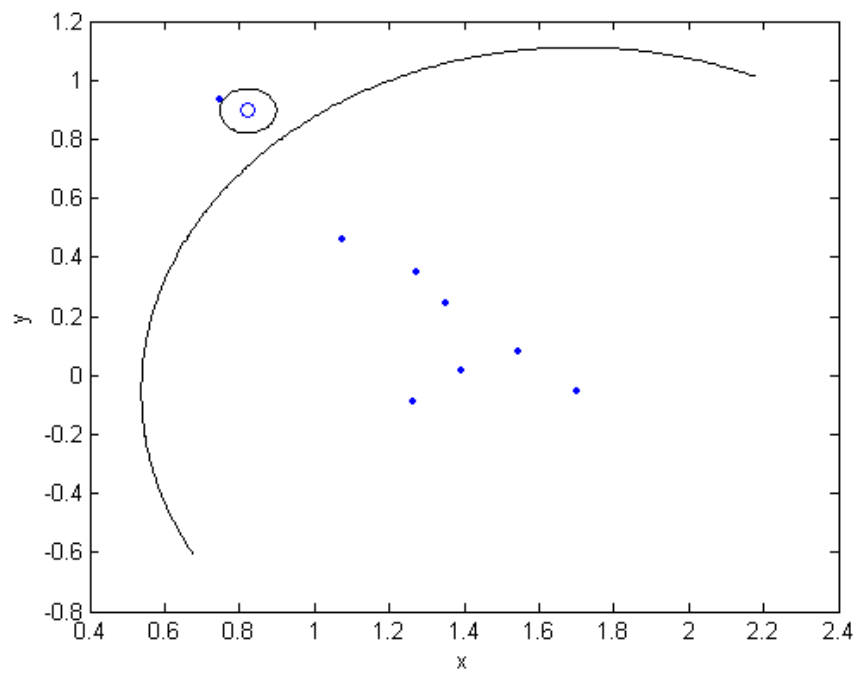


Figure 4c Round-three composite filter

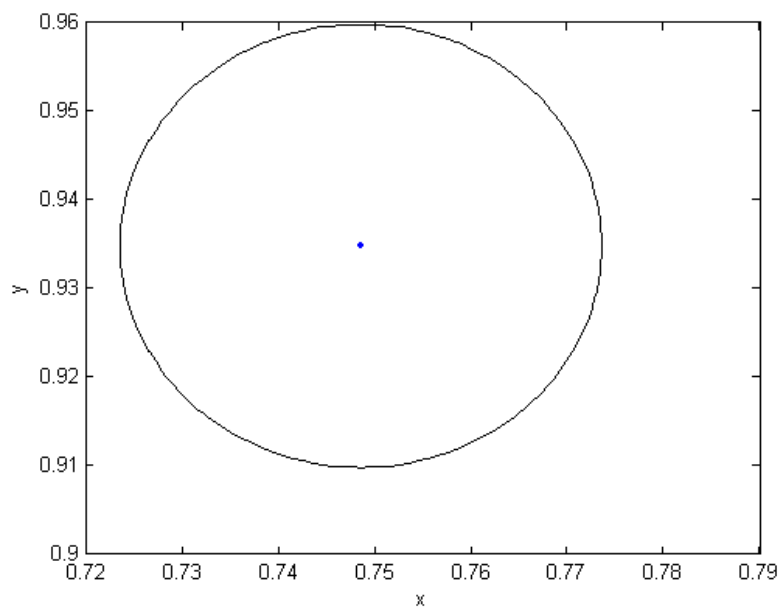


Figure 4d Round-four composite filter

Subsequently ten-thousand new random samples were generated from each distribution function, and the unlabeled samples were presented to the classifier. Each test sample was assigned a class in accordance to the composite-filter ellipse by which it was captured. Test samples that remained free of encapsulation by any of the composite filters remained unclassified. Figure 5 plots the effect of margin-value on classifier performance. As in the previous example, for each setting of the margin-value the simulation was repeated ten times and performance results were averaged. With ten-percent margin one obtains correct classification, misclassification, and indecision rates of 95.6, 2.8, and 1.6 percent, respectively. This performance is remarkable considering the fact that the training process uses only one-hundred labeled samples from each class and no other a priori information about class distributions. Increasing number of trainers to two-hundred per class and using margin value of ten-percent resulted in slight improvement in classifier performance with correct classification, misclassification and indecision rates of 96.4, 2.7, and 0.9 percent, respectively.

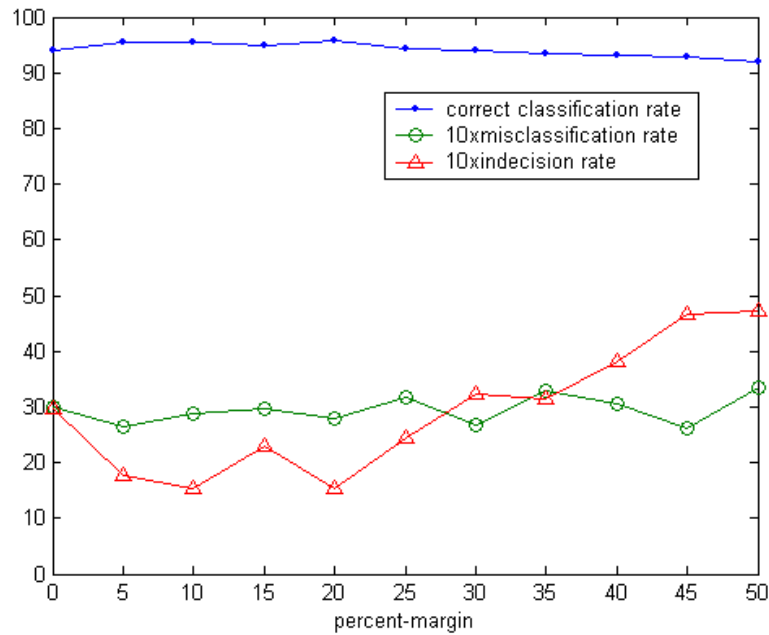


Figure 5: Effect of margin-value of classifier performance

Having complete a-priori information about class distribution functions leads to the optimal classifier as follows

$$\text{test sample} \begin{cases} \text{radial distance to origin} > R_{\text{optimum}} \Rightarrow \text{class} - 2 \\ \text{radial distance to origin} < R_{\text{optimum}} \Rightarrow \text{class} - 1 \end{cases} \quad \text{Eq. 12}$$

Where R_{optimum} is the intersection of two Gaussians with means equal to R_1 and R_2 and standard deviations of σ_{r1} and σ_{r2} .

$$R_{\text{optimum}} = -\frac{1}{2a} \left(b + \sqrt{b^2 - 4ac} \right) \quad \text{Eq. 13a}$$

$$a = \left(\frac{1}{\sigma_{r2}^2} - \frac{1}{\sigma_{r1}^2} \right) \quad \text{Eq. 13b}$$

$$b = 2 \left(\frac{R_1}{\sigma_{r1}^2} - \frac{R_2}{\sigma_{r2}^2} \right) \quad \text{Eq. 13c}$$

$$c = \left(\frac{R_2^2}{\sigma_{r2}^2} - \frac{R_1^2}{\sigma_{r1}^2} + 2 \ln\left(\frac{1}{\sigma_{r1}}\right) - 2 \ln\left(\frac{1}{\sigma_{r2}}\right) \right) \quad \text{Eq. 13d}$$

For the parameter values used in the simulation above one finds $R_{\text{optimum}} = 1.212$. Probability of misclassification is therefore

$$P_e = 2 - \frac{1}{2} \left[\text{erf} \left(\frac{R_{\text{optimum}} - R_2}{\sqrt{2}\sigma_2} \right) - \text{erf} \left(\frac{R_{\text{optimum}} - R_1}{\sqrt{2}\sigma_1} \right) \right] \quad \text{Eq. 14}$$

where erf(x) denotes error function

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad \text{Eq. 15}$$

The optimal detector for the above problem leads to correct classification and misclassification error rates of 98.01 and 1.99-percent, respectively. It is seen that the performance of the margin-setting classifier rivals that of the optimal classifier despite the fact that margin setting is not privy to any a priori information with regards to class distributions. In order to carry out comparisons between this classifier and the one based on Margin-Setting with hyperspheres [10], one-hundred labeled samples from each of the above classes were used to train the hypersphere-based classifier and ten-thousand samples were used to test it. The simulation was repeated ten times and results were

averaged. Two trends were observed. The hyperellipsoid-based classifier, in general, is comprised of a smaller number of composite filters, and its performance is superior. The hypersphere-based classifier in the above example resulted in 94.9, 2.85, and 2.25 percent correct classification, misclassification, and indecision error rates, respectively.

In the next example Class-one and Class-two objects are represented as points in the xy-plane. Polar coordinates (r, ϕ) of each object in the 2D feature space are obtained from the following distribution functions.

$$\begin{aligned}
 \phi_0^{(1)} &\in [-\pi/2, 3\pi/2] \\
 \phi_0^{(2)} &\in [\pi/4, 2.25\pi] \\
 \phi^{(1)} &\sim N(\phi_0^{(1)}, (0.1\phi_0^{(1)})^2) \\
 \phi^{(2)} &\sim N(\phi_0^{(2)}, (0.1\phi_0^{(2)})^2) \\
 r_0^{(1)} &= \phi^{(1)} + \pi/2 \\
 r_0^{(2)} &= \phi^{(2)} \\
 r^{(1)} &\sim N(r_0^{(1)}, (0.1r_0^{(1)})^2) \\
 \rho^{(2)} &\sim N(r_0^{(2)}, (0.1r_0^{(2)})^2)
 \end{aligned}
 \tag{Eq. 16}$$

Where superscripts represent class designations and zero-subscript denotes mean value. A Class-one object, for example, is obtained by random selection of $\phi_0^{(1)}$ from the above uniform distribution function, and subsequent selection of $\phi^{(1)}$ from the normal distribution function with mean and standard deviation equal to $\phi_0^{(1)}$, and $0.1\phi_0^{(1)}$, respectively. The selected $\phi^{(1)}$ is used to compute $r_0^{(1)}$, which is then used as mean of the normal distribution from which $r^{(1)}$ is obtained. Figure 6 shows one-hundred trainers for each of the two classes.

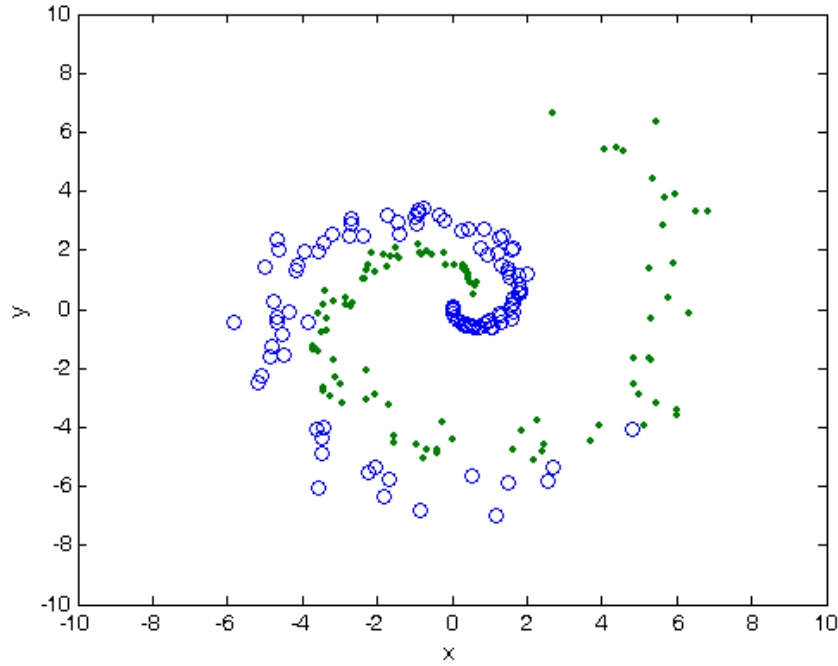


Figure 6: Class-one (circles) and Class-two (dots) trainers

The set of exemplars in Figure 6 was used to train a classifier. Training with ten-percent margin resulted in an eight round classifier. Figure 7a-h show composite filters, each comprised of two ellipses, representing ordered classifier rounds. To label an untrained-on object it is tested with the round-one filter, and is labeled with the respective class if it falls inside one of the two ellipses, and the process is terminated. If not, it is passed to and is tested with the round-two filter, and so on. If the object falls inside both ellipses of a particular composite filter it is assigned a class label according to its proximity factor which is defined in (10).

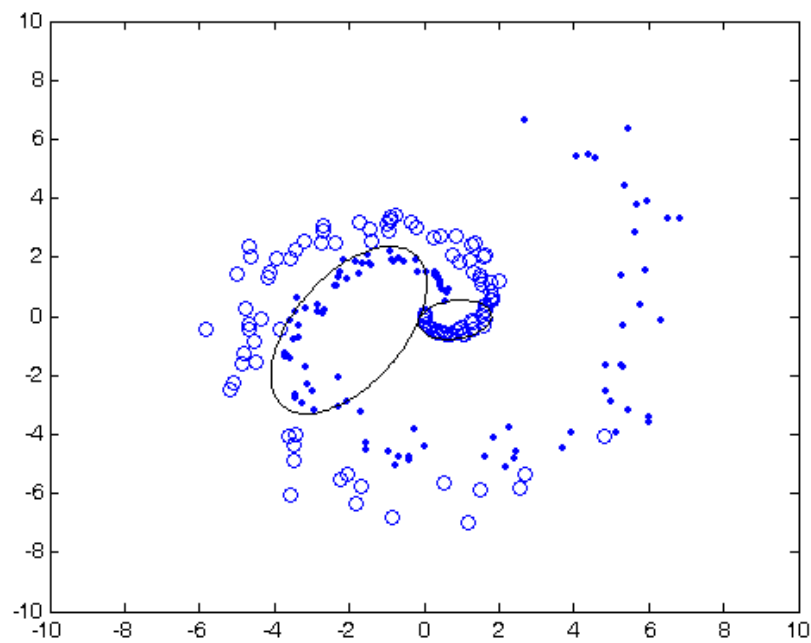


Figure 7a: Round-one composite filter

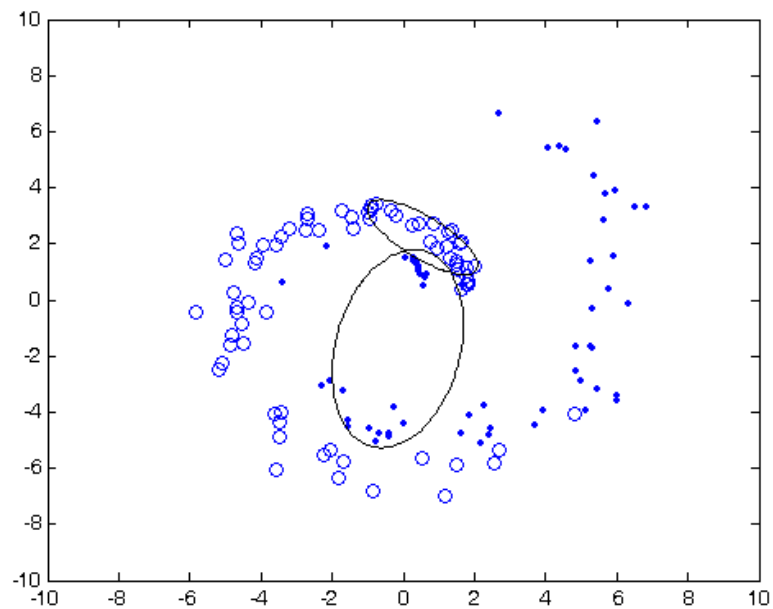


Figure 7b: Round-two composite filter

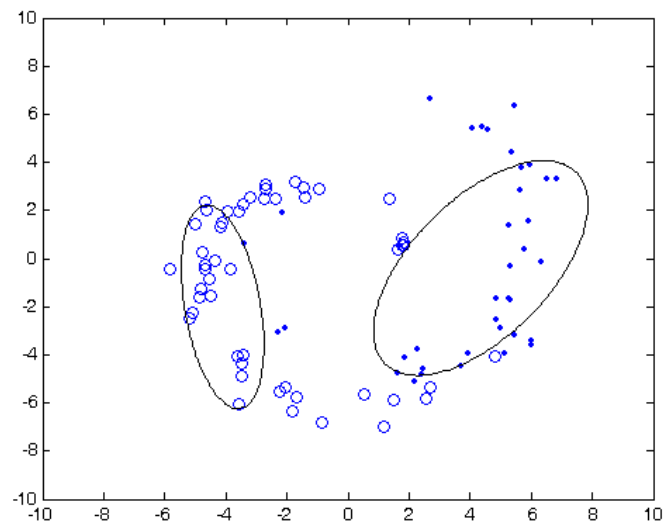


Figure 7c Round-three composite filter

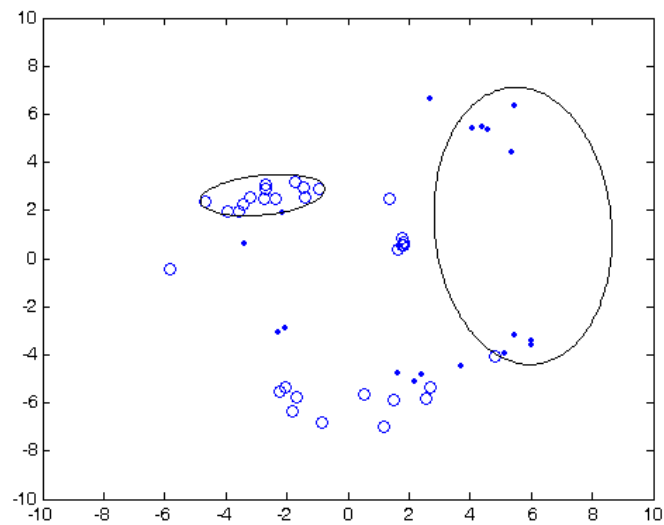


Figure 7d Round-four composite filter

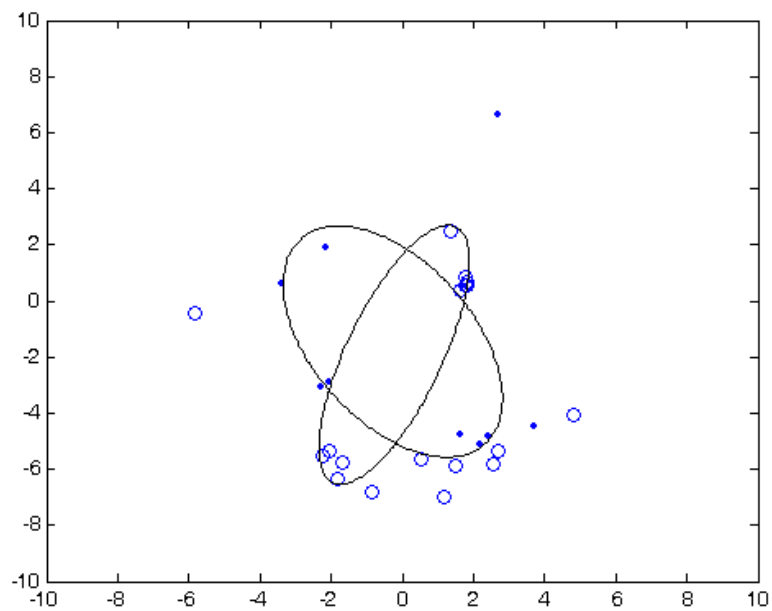


Figure 7e Round-five composite filter

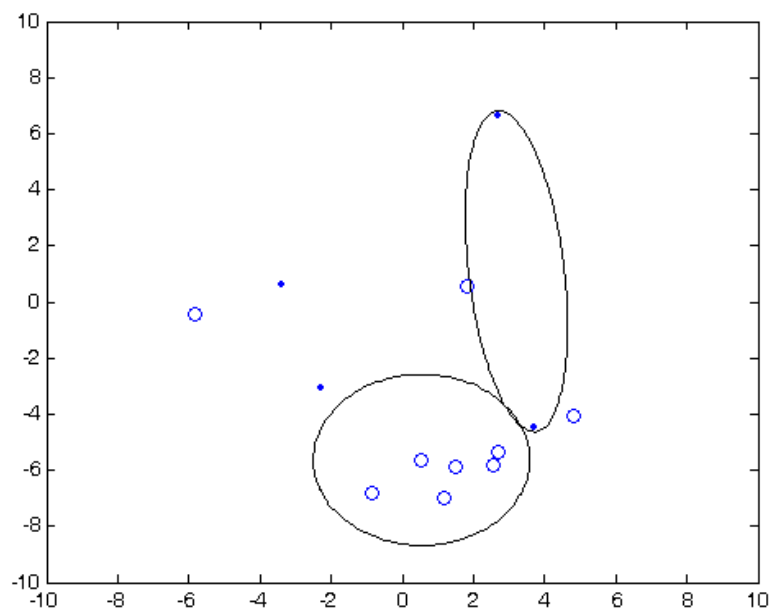


Figure 7f Round-six composite filter.

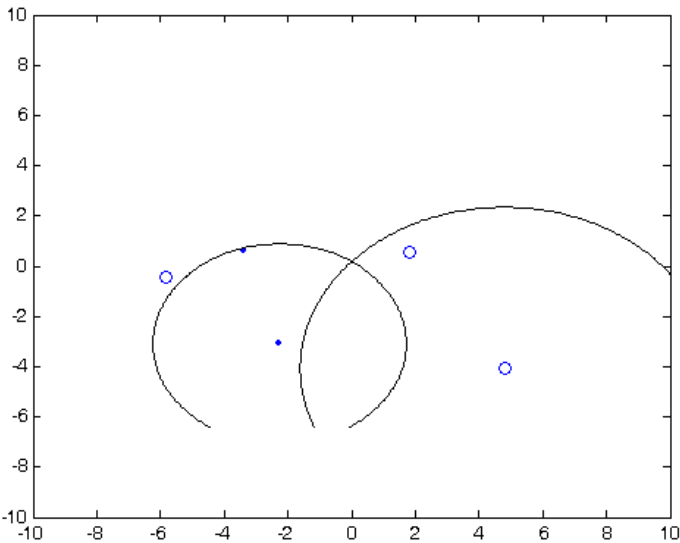


Figure 7g Round-seven composite filter

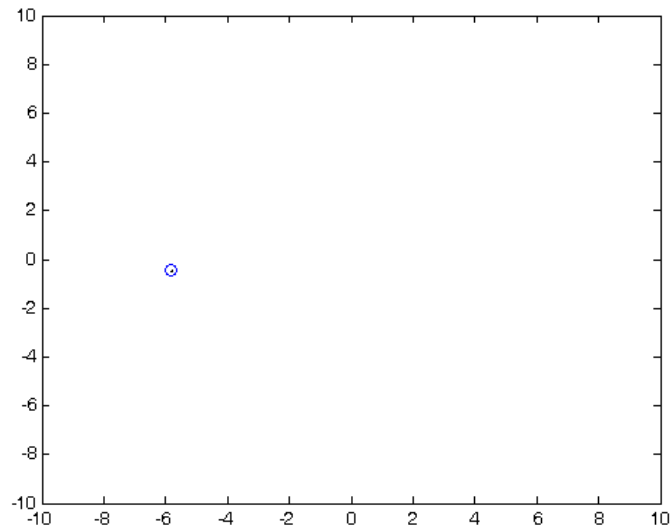


Figure 7h Round-eight composite filter

The classifier was tested by using it to label one-thousand randomly generated objects from each of the two above distributions. In order to obtain statistically reliable performance data, for each margin value, the simulation was repeated ten times and results were averaged. Each simulation consisted of generating one-hundred objects from each distribution, using the labeled objects to train a classifier, generating one-thousand random test objects from each distribution, utilizing the classifier to label the untrained-on test objects, and comparing the assigned labels to actual class dispositions. Plots of Figures 8a-c show classifier performance as function of margin value. Performance data for classifiers with hyperspherical decision surfaces (circles in 2D) are also plotted in the same figures. Results of this example show that hyperellipsoidal decision surfaces lead to superior classifiers compared to the hyperspherical based classifiers. For example, a hyperellipsoidal classifier trained with thirty-percent margin results in correct classification, misclassification, and indecision rates of 94.308, 2.3075, 3.3845-percent, respectively, whereas the respective results for a hyperspherical classifier with the same margin are 82.57, 1.765, 15.665. It is seen, in this example, that for the hyperellipsoidal classifier misclassification error rates can be reduced without leading to inordinate increase in indecision rates.

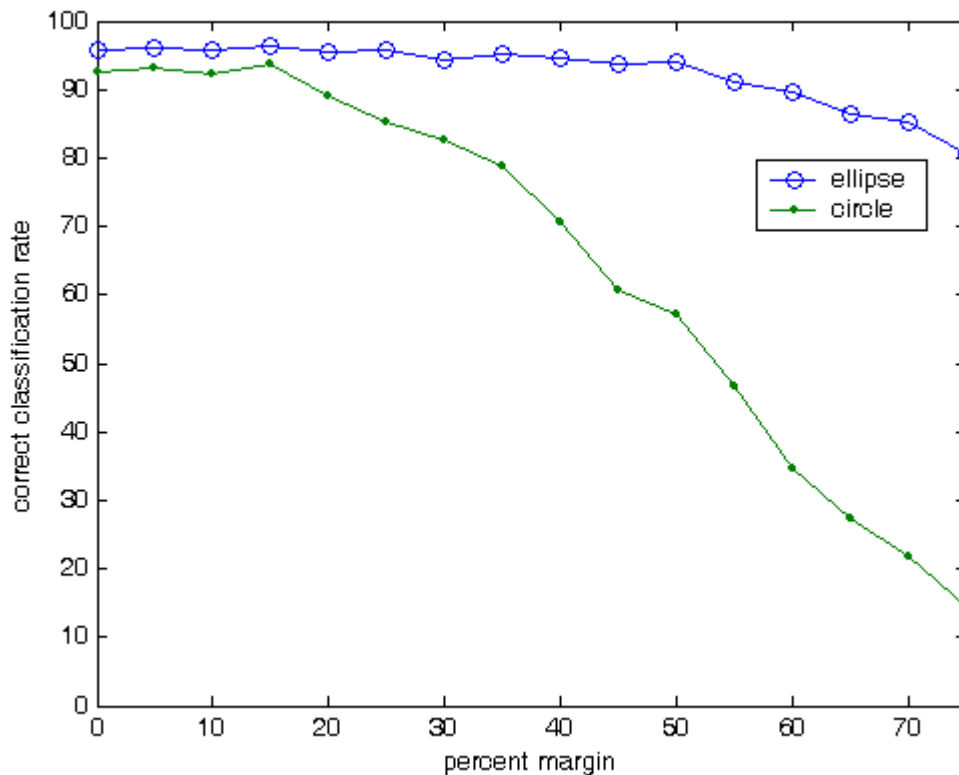


Figure 8a: Correct classification rate as function of margin

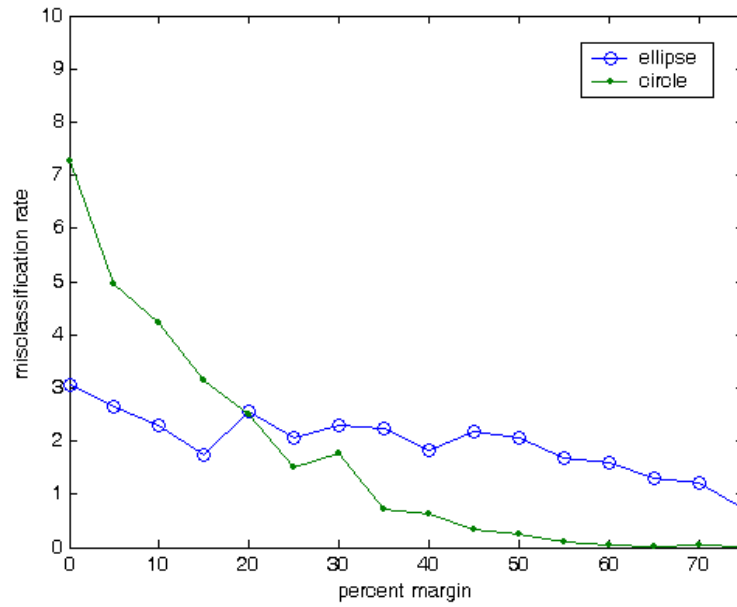


Figure 8b Misclassification error rate as function of margin

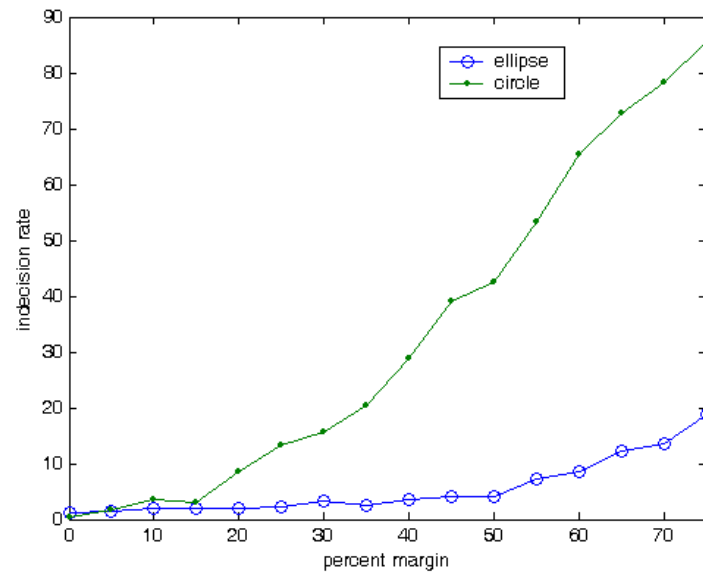


Figure 8c Indecision rate as function of margin

Conclusions

The switch from hyperspheres in earlier cases of Margin Setting to hyperellipsoids makes this classification method outstanding in three respects:

1. Very few fair trainers from each class are needed, compared to what PAC Learning suggest, because we have abandoned the underlying assumption that all tasks must be done with either a single strong classifier or many weak one. New paradigms lead to new results.
2. Margin-Setting with hyperellipsoids outperforms Margin-Setting with hyperspheres in every quantitative comparison shown here.
3. The resulting classification appears to be very close to the best possible, in the examples shown here, despite using only a few training samples.

Application of Fuzzy Clustering to Color Image Segmentation

This section describes an algorithm that segments a color image into a user-specified number of classes (M) in accordance with class association of its pixels. It uses N (user-specified) randomly selected pixels from the input image (trainers) and computes M prototypes (M is number of classes) using Fuzzy Clustering described in the appendix below. The input trainers are unlabeled.

In order to obtain quantitative results for the Fuzzy Clustering algorithm the following tests were conducted. Following these tests application of the algorithm to color image segmentation is presented. The mathematical formulation of the algorithm is given in the Appendix at the end of Section II.

We started with two Gaussian distributions in the 2D-space. It is assumed that x and y components of each sample point are obtained from independent distributions with equal variances. In the first example Classes A and B are obtained from two 2D distributions with equal variances $\sigma_A = \sigma_B$. Equal numbers of randomly generated trainers from each class were combined and were utilized as unlabeled trainer.

Fuzzy clustering was then applied to the set of unlabeled trainers to evolve two prototypes. A large number of test points were then generated from the distribution functions described above. The test points were then classified in accordance to their Euclidean distances to the two prototypes.

Figure 9 shows classification error rate as function of separation factor with number of trainers (unlabeled) utilized from each class as parameter. It is noted that separation factor (SF) is defined below.

$$SF = \frac{\text{Euclidean distance between class - means}}{\sqrt{\sigma_A^2 + \sigma_B^2}} \quad \text{Eq. 17}$$

For each case number of trainers was kept fixed and separation-factor was varied from 0.25 to 4. Figure 9 shows that for a fixed number of trainers, error rate decreases with increasing means-separation, as expected. It is also noted that larger number of trainers results in lower error rates. The effect of number of trainers on classification error rate, however, is very small.

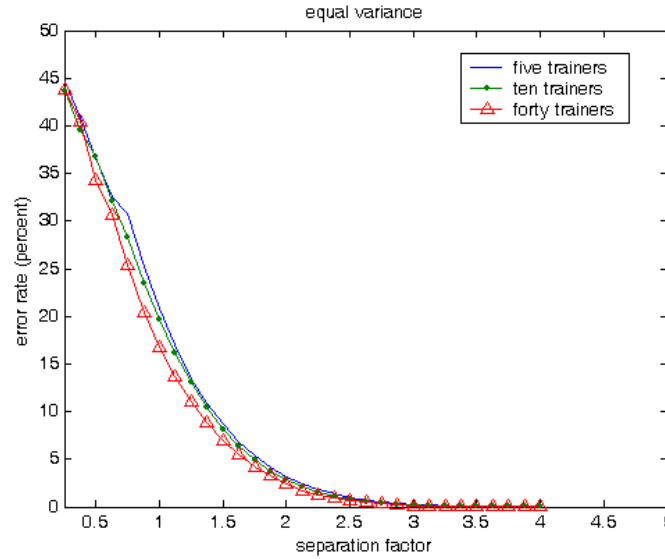


Figure 9: Classification error rate as function of separation-factor. Two Gaussian distributions have equal standard deviations

The example of Figure 10 shows classification error rates when two distributions have unequal variances ($\sigma_B=2\sigma_A$). Similar trends are observed with the exception that in this case number of trainers has an slightly more pronounced effect on classification error rate for lower values of mean-separation-factor.

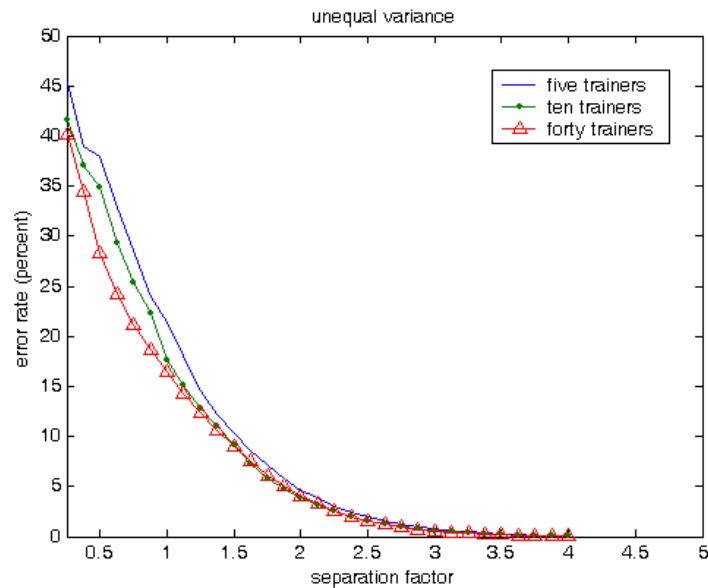


Figure 10: Classification error rate as function of separation-factor. Two Gaussian distributions have unequal standard deviations. Variance of B is four times greater than variance of A.

The example of Figure 11 shows classification error rates when the disparity between distribution function is even greater ($\sigma_B=4\sigma_A$). It is noted that error rates are in general slightly lower, with respect to previous cases, using same parameter values (separation factor and number of trainers).

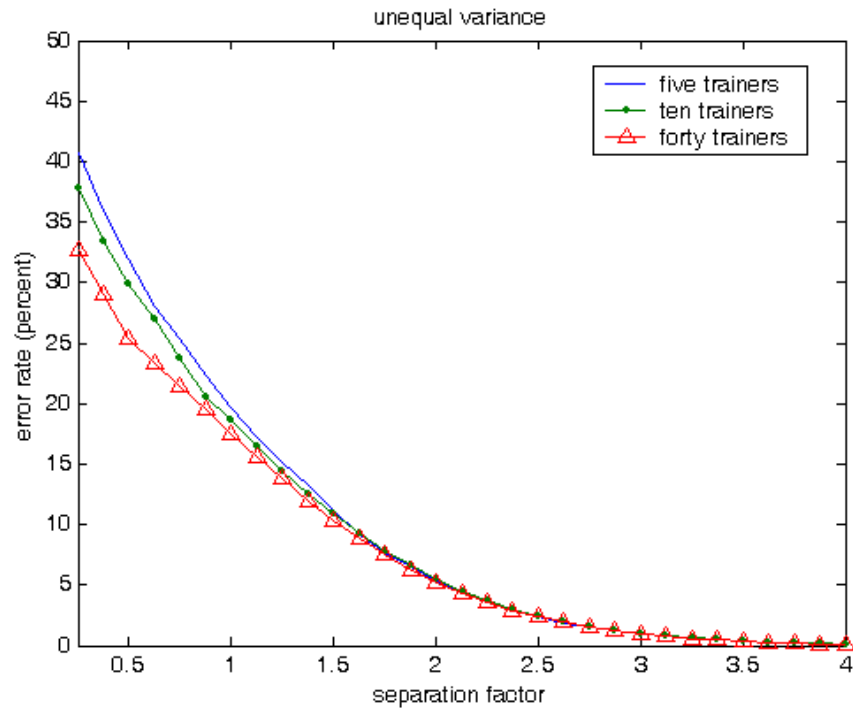


Figure 11: Classification error rate as function of separation-factor. Two Gaussian distributions have unequal standard deviations. Variance of B is sixteen times greater than variance of A.

The example of Figure 12a shows the input image (upper-left), and the result of color segmentation. In this example one-hundred pixels ($N=100$) were randomly selected from the input image (Figure 12b) and were used as trainers. It is noted that the trainers bear no labels. The algorithm was then tasked to partition the training set into three classes ($M=3$). Figure 12c shows the evolution of class prototypes. It is noted that all three prototypes are initially close to each other and to the centroids of the training set. The prototypes migrate towards their true positions and true prototypes are evolved as shown in Figure 12c. In this example it took twenty iterations for all three prototypes to reach their final destinations.

After computation of prototypes, the image is segmented in accordance to pixel-prototype distance. In the examples presented here Euclidean distance is used. Pixels are labeled with the class associated with the closest prototype to the pixel. In examples to be presented later Mahanobis distance is used to determine class association of each pixel.

Table 4: Initial and final prototype vectors for three color classes

	Initial Prototypes			Final Prototypes		
	R	G	B	R	G	B
Class-one	0.8527	0.6780	0.4919	0.5662	0.2550	0.2334
Class-two	0.8613	0.6866	0.4917	0.9455	0.7160	0.3445
Class-three	0.8795	0.7002	0.5073	0.8849	0.7567	0.6226



Figure 12a: Original image (upper-left), and color-segmented images. Classes-one (upper right), two (lower-left), and three (lower-right).

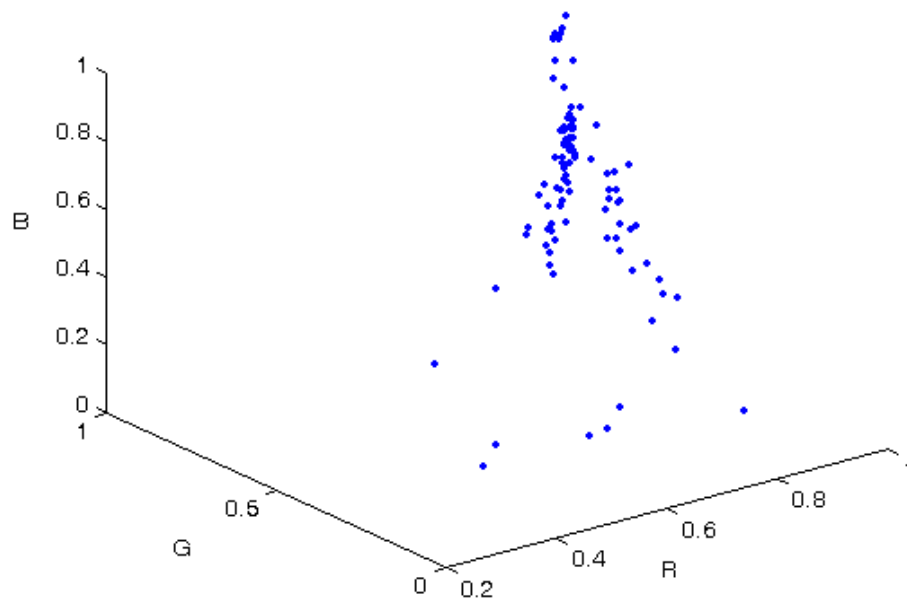


Figure 12b Training set comprised of one hundred randomly selected pixels (unlabeled) from the original image (upper-left of Figure 12a).

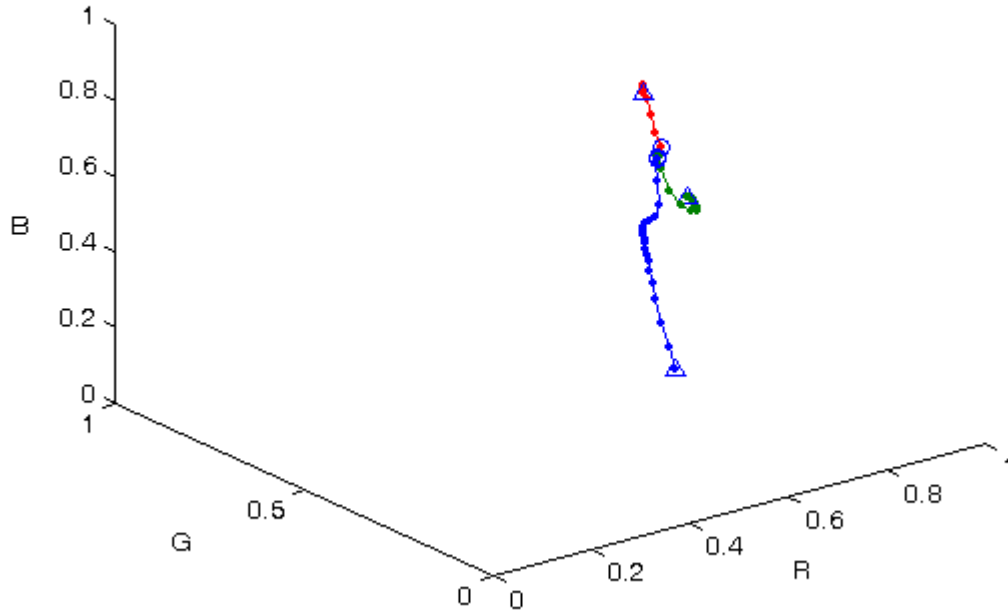


Figure 12c Evolutions of prototypes for classes-one through three are shown in blue, green, and red, respectively. Initial estimates of prototypes are denoted as circles and final estimates are triangles.

In the next example a Mondrian painting is sampled and one-hundred randomly picked unlabeled pixels are used as trainers. The original image and training samples are shown in Figures 13a, c, respectively. The training set was then partitioned into four classes using fuzzy clustering. Figure 13d shows the evolution of class prototypes. Figure 13b shows the segmentation result, where each image represents preserving pixels, in the input image, that are members of respective class.

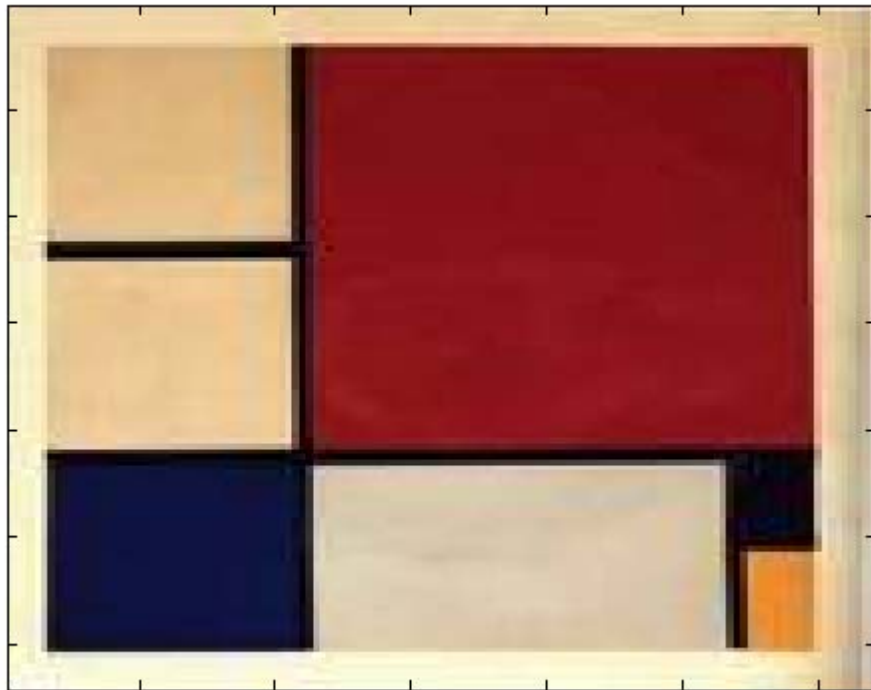


Figure 13a: Mondrian painting

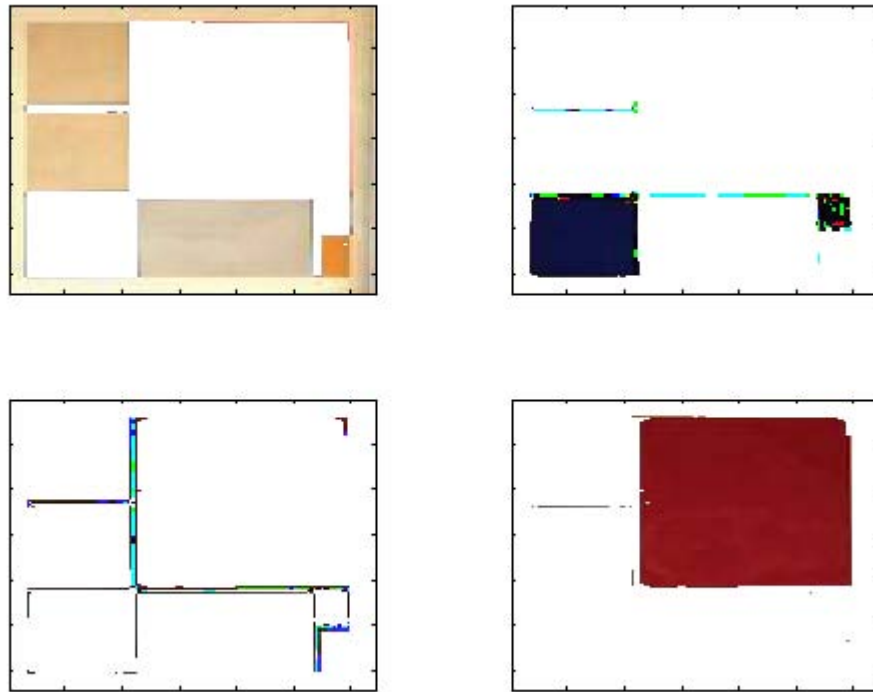


Figure 13b Segmented image

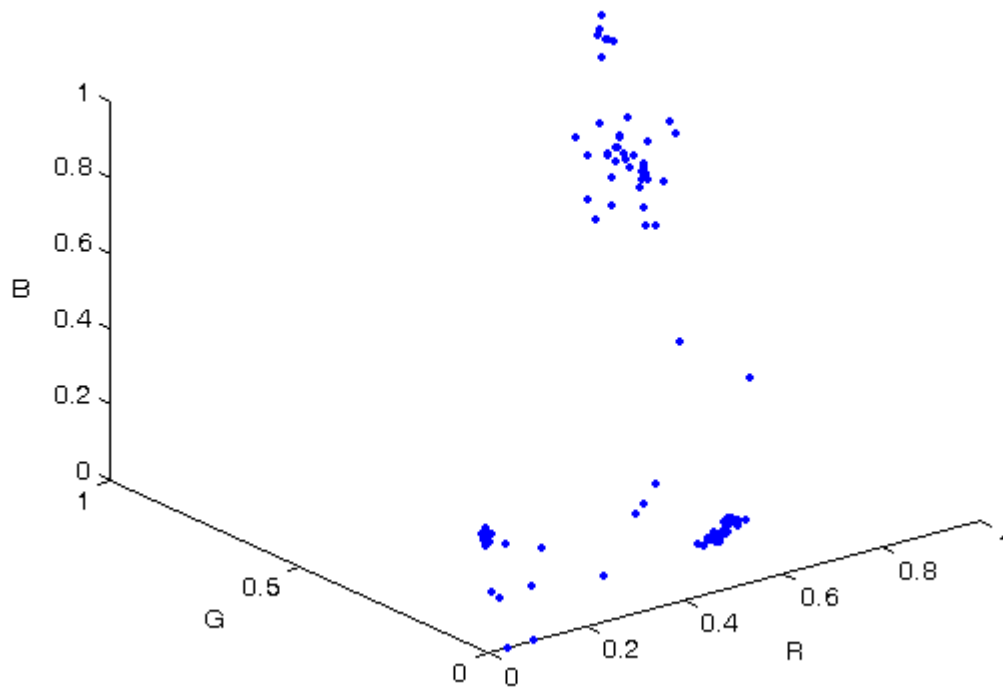


Figure 13c Training set comprised of one-hundred randomly selected pixels (unlabeled) from the original image (Figure 13a).

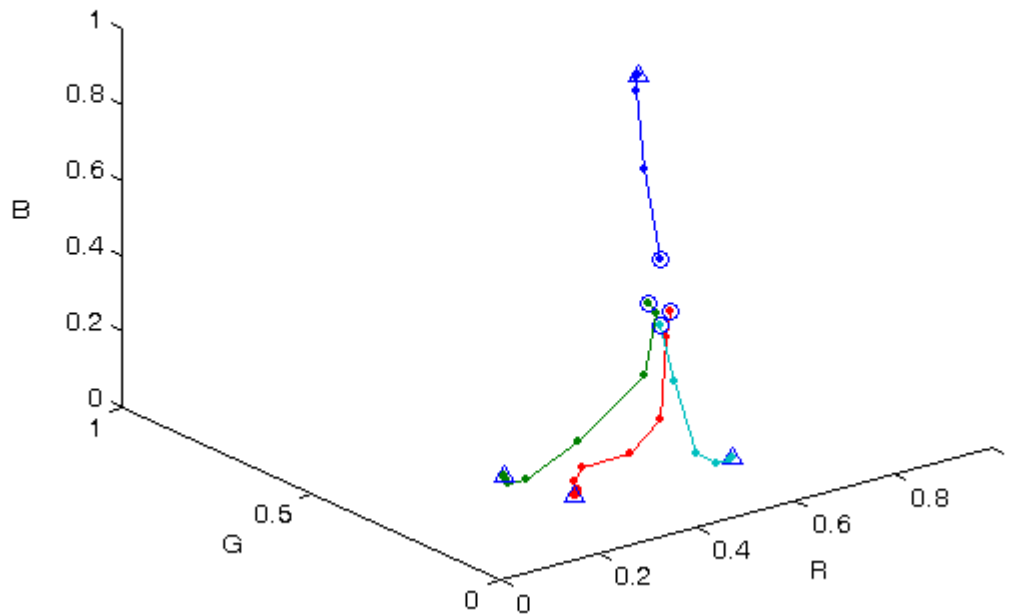


Figure 13d Evolutions of prototypes for classes-one through four are shown in blue, dark-green, and red, and light-green, respectively. Initial estimates of prototypes are denoted as circles and final estimates are triangles.

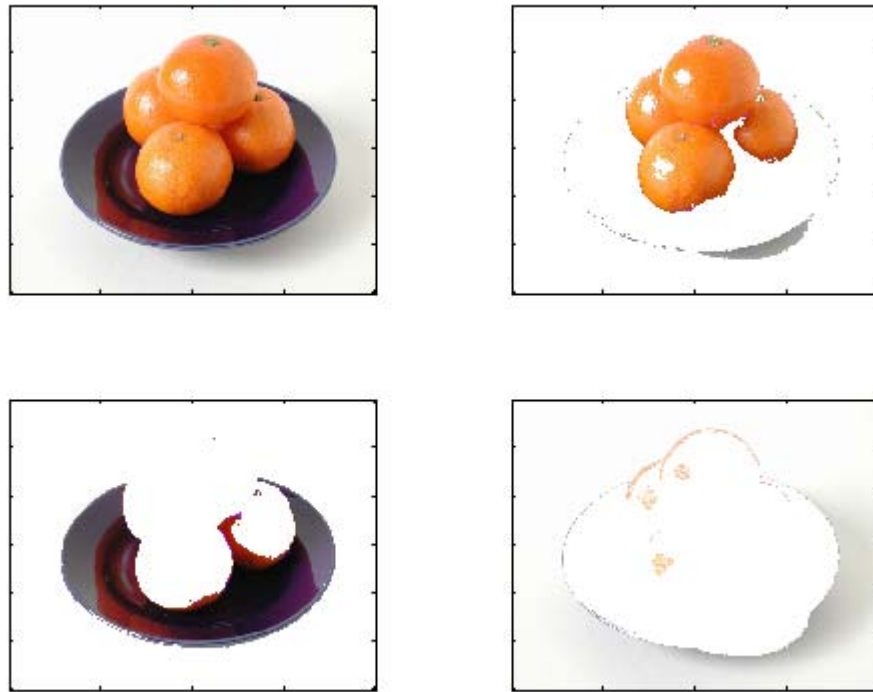


Figure 14a: Upper left shows the input image. Input image is filtered using a three-class filter.

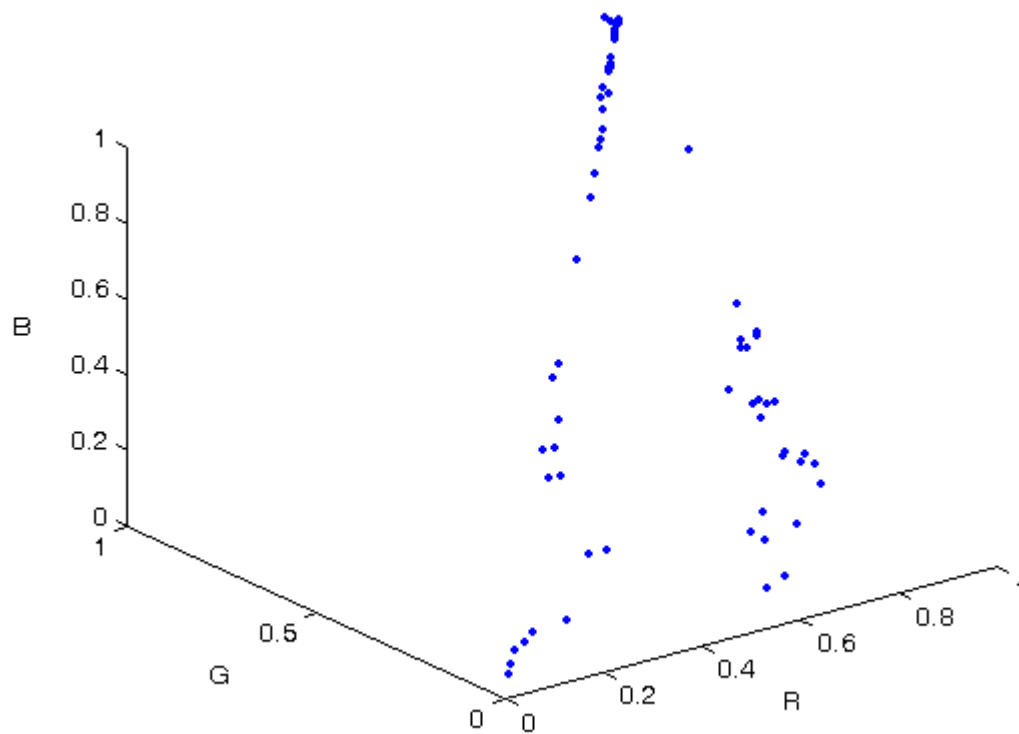


Figure 14b Training set comprised of one-hundred randomly selected pixels (unlabeled) from the input image (Figure 14a).

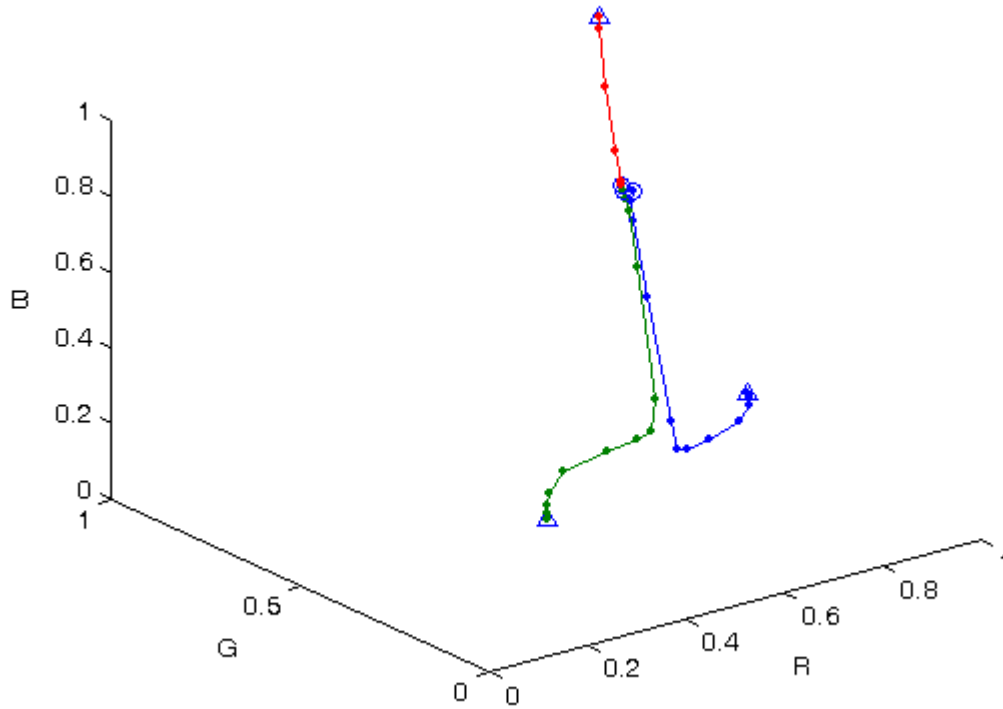


Figure 14c Evolutions of prototypes for classes-one through three are shown in blue, green, and red, respectively. Initial estimates of prototypes are denoted as circles and final estimates are triangles.

Mathematical Formulation

Given a set of N data points in M -dimensional space, and a user-specified integer representing number of clusters (classes) Q , the algorithm described here computes a set of Q prototypes and a $Q \times N$ membership matrix. Each prototype is a vector in M -space and is the optimal representation of the corresponding class. Each element of the membership matrix represents the degree of membership (association) of a data points in the respective cluster.

$$\begin{aligned}
 \mathbf{X} &= \{X_n : 1 \leq n \leq N\} \\
 X_n &= \left\{ x_{mn} : 1 \leq m \leq M, 1 \leq n \leq N \right\} \\
 \mathbf{Y} &= \{Y_q : 1 \leq q \leq Q\} \\
 Y_q &= \left\{ y_{mq} : 1 \leq m \leq M, 1 \leq q \leq Q \right\}
 \end{aligned}
 \tag{Eq. 18}$$

$$x_{mn}, y_{mq} \in \mathfrak{R}$$

Where \mathbf{X} , \mathbf{Y} represent, respectively, the set of data points and prototype vectors (in M-space), and \mathfrak{R} is the set of real numbers. Our objective is to utilize data points in (18) in order to partition M-space into Q distinct regions with each region represented by a prototype vector Y_q . Each original data point will be linked to all Q regions with varying degrees of association determined by elements of the membership matrix. We will describe an iterative algorithm for computation of the prototype vectors. The prototype vectors are then used to make hard decisions with regard to new input data points. A new data point is associated with the prototype (class) to which it is closest in accordance to some predefined distance measure.

$$\begin{aligned}
 S &= [s_{qn}] \quad , 1 \leq q \leq Q, \quad 1 \leq n \leq N \\
 s_{qn} &= \frac{\alpha_n}{\left[\sum_{p=1}^Q \left(d_{qn} / d_{pn} \right)^{\frac{2}{(u-1)}} \right]} \\
 \alpha_n &= \frac{1}{\sum_{q=1}^Q \frac{1}{\sum_{p=1}^Q \left(d_{qn} / d_{pn} \right)^{\frac{2}{(u-1)}}}} \\
 d_{pn} &= \|Y_p - X_n\|
 \end{aligned}
 \tag{Eq. 19}$$

Where S is the membership (association) matrix, s_{qn} is the degree with which data point- n is associated with (member of) cluster- q , and d_{qn} is the distance between data point- n and prototype- q . Here Euclidean distance is used as a measure of distance between vectors in M-space. The exponent parameter $u \in \{[1, \infty]\}$ is user-specified and determines the fuzziness of the clustering process. It is noted from (19) that the membership matrix is normalized such that sum of each column is equal to one. When $u = \infty$, each data point belongs to all clusters uniformly and $s_{qn} = 1/Q$,

$1 \leq n \leq N, \quad 1 \leq q \leq Q$. When $u = 1$, however, clustering is not fuzzy and each data point

is associated with a unique cluster $s_{qn} = 1$, $d_{qn} < d_{pn} \quad \forall p \neq q$ and $s_{qn} = 0$ otherwise. In hard clustering, $u = 1$, each column of S contains a single one and the rest of entries for that column are zero. Typical value for u is 2.5.

The process starts with generating a random membership matrix, called the zero-order membership matrix $S^{(0)}$. Matrix elements are chosen from a uniform probability distribution function $s_{qn}^{(0)} \in [0,1]$. The matrix is then normalized with respect to sum of each column. The randomly generated membership matrix is then used to compute Q zero-order prototype vectors, one for each cluster. A particular prototype vector is computed as the weighted sum of the entire set of data points, where each data point is weighted in accordance to its association to (membership in) the respective cluster.

$$Y^{(0)} = \{Y_q^{(0)} : 1 \leq q \leq Q\}$$

$$Y_q^{(0)} = \frac{\sum_{n=1}^N (s_{qn}^{(0)})^u X_n}{\sum_{n=1}^N (s_{qn}^{(0)})^u} \quad ; \quad 1 \leq q \leq Q \quad \text{Eq. 20}$$

Where $Y_q^{(0)}$ is the cluster- q zero-order prototype vector, X_n is the n th data vector, $s_{qn}^{(0)}$ ($1 \leq q \leq Q$, $1 \leq n \leq N$) are elements of the randomly generated zero-order membership matrix, and u is the user-specified exponential parameter. The zero-order prototype vectors are then utilized to compute the first-order membership matrix.

$$S^{(1)} = [s_{qn}^{(1)}] \quad , 1 \leq q \leq Q, \quad 1 \leq n \leq N$$

$$s_{qn}^{(1)} = \frac{\alpha_n^{(0)}}{\left[\sum_{p=1}^Q \left(d_{qn}^{(0)} / d_{pn}^{(0)} \right)^{\frac{2}{(u-1)}} \right]}$$

$$\alpha_n^{(0)} = \frac{1}{\sum_{q=1}^Q \frac{1}{\sum_{p=1}^Q \left(d_{qn}^{(0)} / d_{pn}^{(0)} \right)^{\frac{2}{(u-1)}}}} \quad \text{Eq. 21}$$

$$d_{pn}^{(0)} = \|Y_p^{(0)} - X_n\|$$

$$G^{(1)} = [g_{qn}^{(1)}] \quad ; \quad g_{qn}^{(1)} = |s_{qn}^{(1)} - s_{qn}^{(0)}|$$

$$\delta^{(1)} = \max_{q,n} (g_{qn}^{(1)})$$

Where $S^{(1)}$, and $G^{(1)}$ denote, respectively, first-order membership and gradient matrices, and $\delta^{(1)}$ is the first-order gradient. The first-order membership matrix is then used to compute the first-order prototype vectors, which are subsequently used to compute the second-order membership matrix and gradient. The iterative process continues until a stopping criterion is met. Stopping criterion may be maximum number of iterations (orders), in which case, the process stops after the number of iterations is reached. One may also use gradient as the stopping criterion. In this case the process is terminated when gradient falls below a user-specified threshold i.e. $\delta^{(r)} < T = 0.001$.

Simulation Results

A set of one-hundred two-dimensional data points was generated from two Gaussian probability density functions. Random variables representing x, y components of each data point are independent. The normal distribution functions have means at (1,3) and (-2,-1), and standard deviations along x, y directions for both distributions are equal to 1. Figure 9 shows the input data points, where circles and stars represent vectors obtained from first and second distributions, respectively. These vectors were used as input data (without labels) with exponent parameter $u=2$, number of clusters $Q=2$ and stop condition, $T = 0.001$. The iteration process converged (value of gradient δ in equation-21 dropped below the threshold) after ten rounds. Figure 9 shows that zero-order prototype vectors are close to each other and are located roughly at the center of gravity (centroid) of the entire data set. This is to be expected, as all data points are initially assumed to belong to both clusters with equal probabilities. It is also seen that as iterations are performed the cluster prototypes migrate towards their true locations. Elements of the membership matrix are plotted in Figure 10. It is seen that the first fifty data points (circles) are more strongly associated with cluster-one, and the second fifty data points (stars) are associated more with cluster-2 as expected and seen from Figure 15.

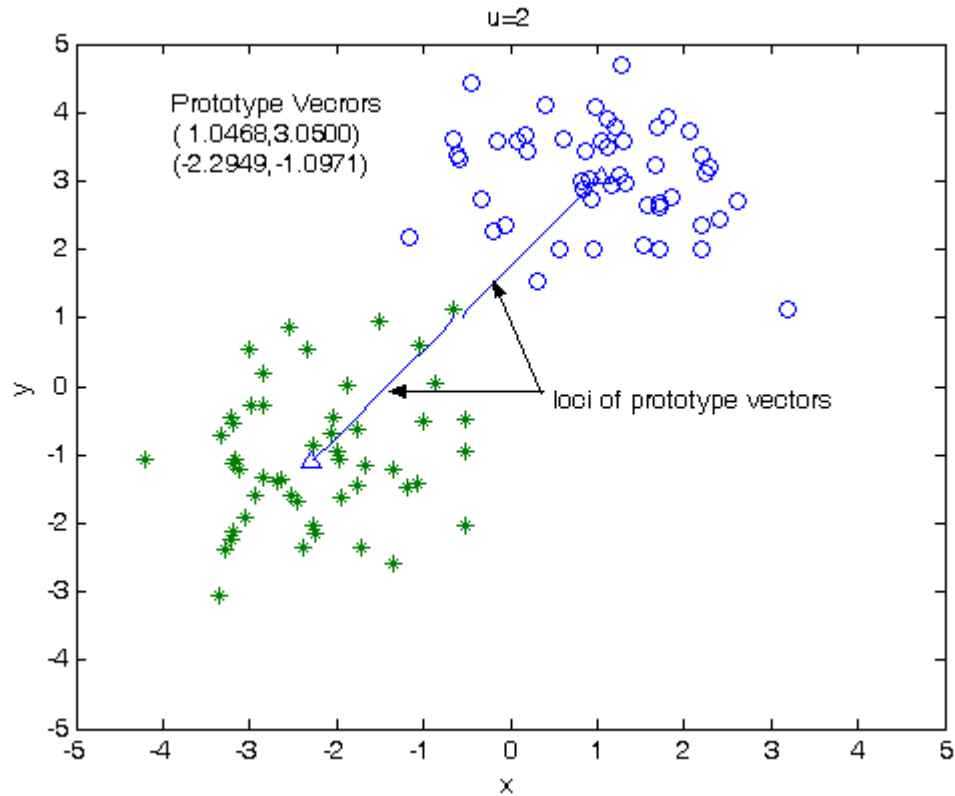


Figure 15: Data vectors (circles and stars) and prototype vectors (triangles) for a two-cluster problem in 2D space.

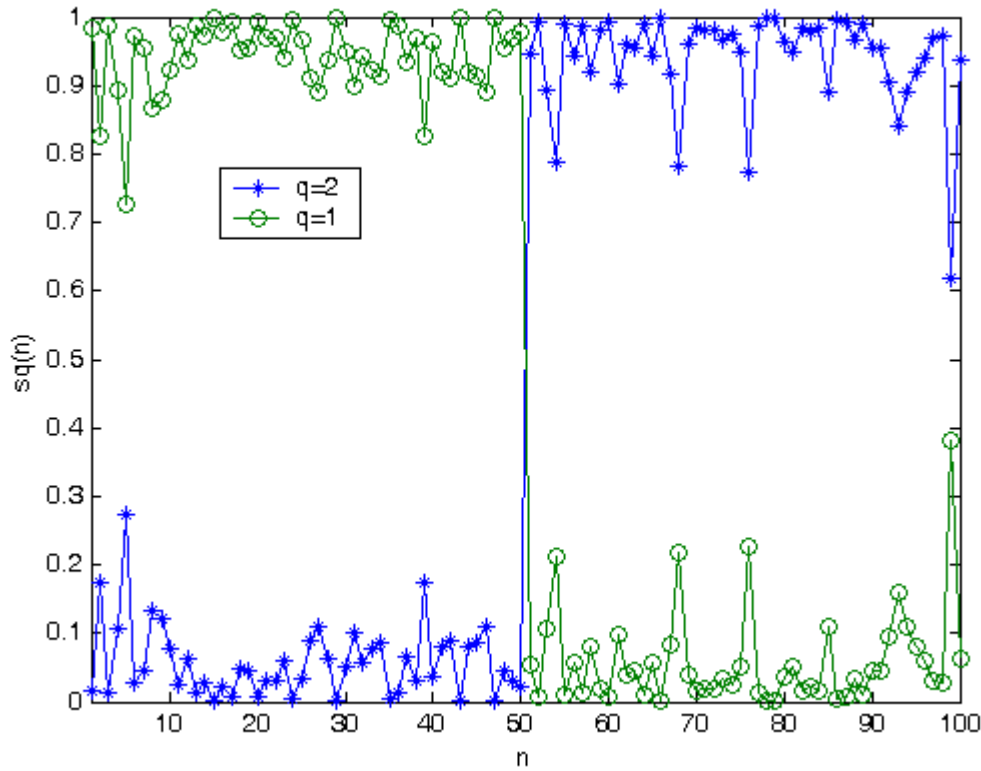


Figure 16: Elements of the two-row membership matrix S . First row is shown with circles and second row with stars. $Sq(n)$ is degree of membership of n th data point in q th cluster.

In the example of Figure 17 a set of forty 2D vectors comprised of twenty-five and fifteen vectors obtained from two normal distributions centered at (1,3) and (-2,-1), respectively, are used as input data points. As before (x, y) components of each vector are independent Gaussian random variables with standard deviations of (0.5,1), and (1,2), for the first and second distributions, respectively. The simulation was executed with parameters $Q=1$, $u=2$, and $T=0.001$. As expected, both prototype vectors were initially close to the centroids of the entire data set and migrated to their final positions after ten iteration rounds. Figure 12 plots the membership coefficients (elements of membership matrix), and shows that the first twenty-five points (circles in Figure 17) are associated more strongly with the first prototype vector (0.8899,2.9917). Three of the last fifteen data points, however, are associated more strongly with the first prototype vector compared to the second (-2.2643, -1.6864).

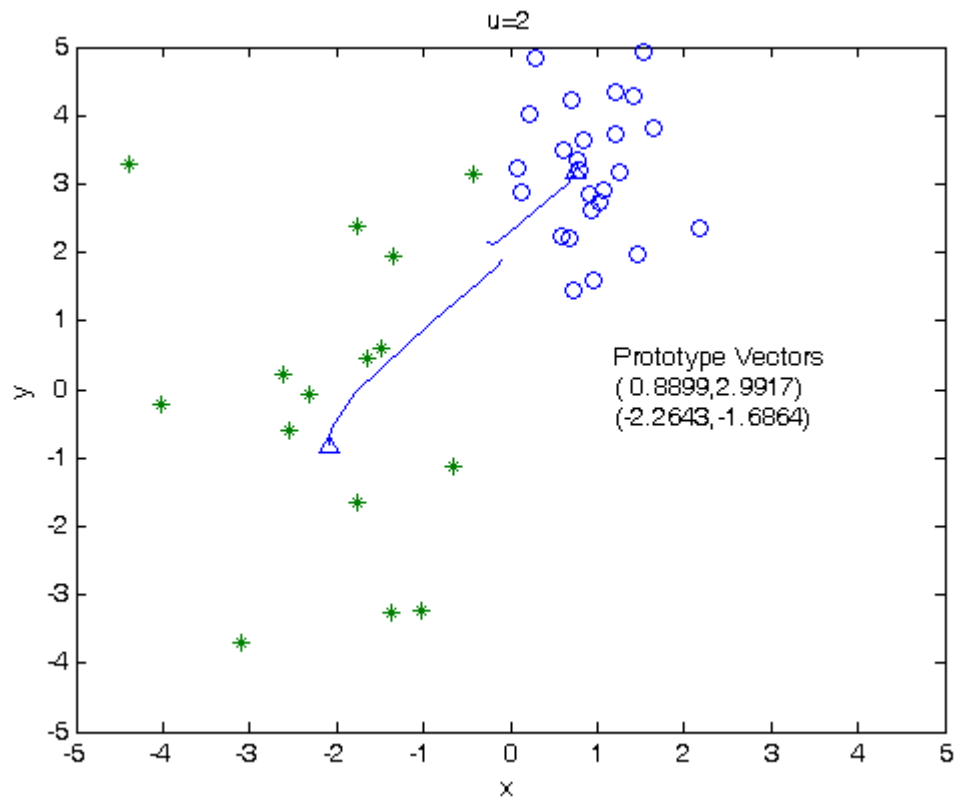


Figure 17: Data vectors (circles and stars) and prototype vectors (triangles) for a two-cluster problem in 2D space.

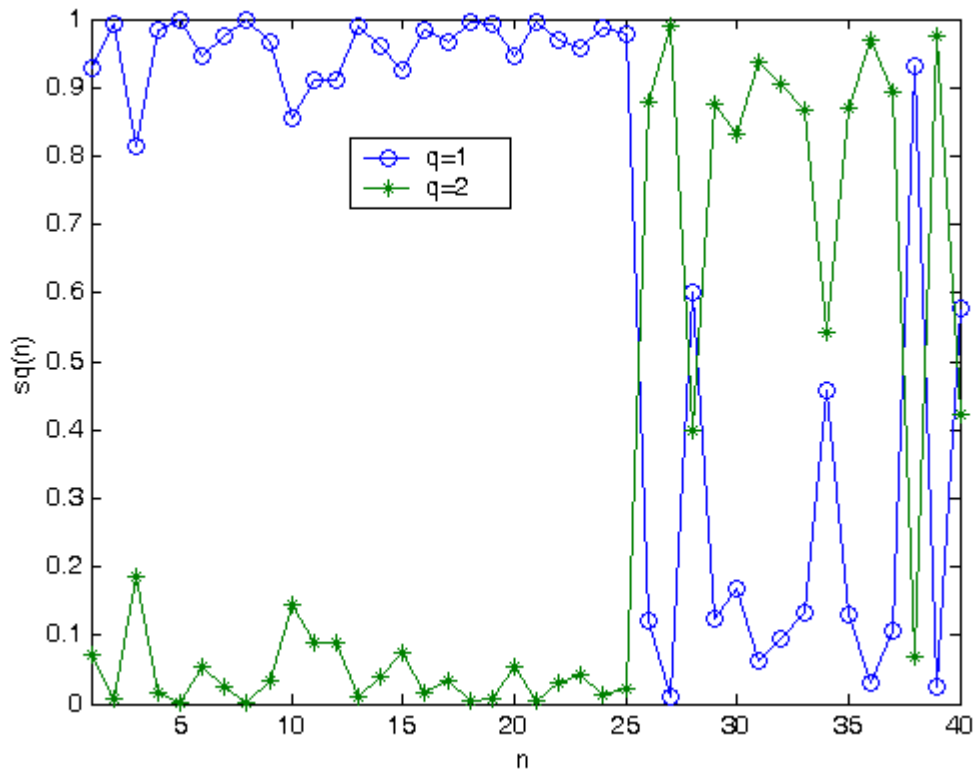


Figure 18: Elements of the two-row membership matrix S . First row is shown with circles and second row with stars. $Sq(n)$ is degree of membership of n th data point in q th cluster.

In the example of Figure 19 we obtain 125 data points in 2D space from three random distributions centered at (1,3), (-2,-6), and (3,-2) and shown as circle (fifty), star (twenty-five), and dot (fifty). The clustering algorithm was executed using $Q=3$, $u=2$, $T=0.001$. The process converged after seventeen iterations. The computed prototype vectors (triangles), as well as initial values and paths of progression for each are shown in the figure. It is noted that prototype vectors for all three classes start close to each other and at roughly the centroids of data points.

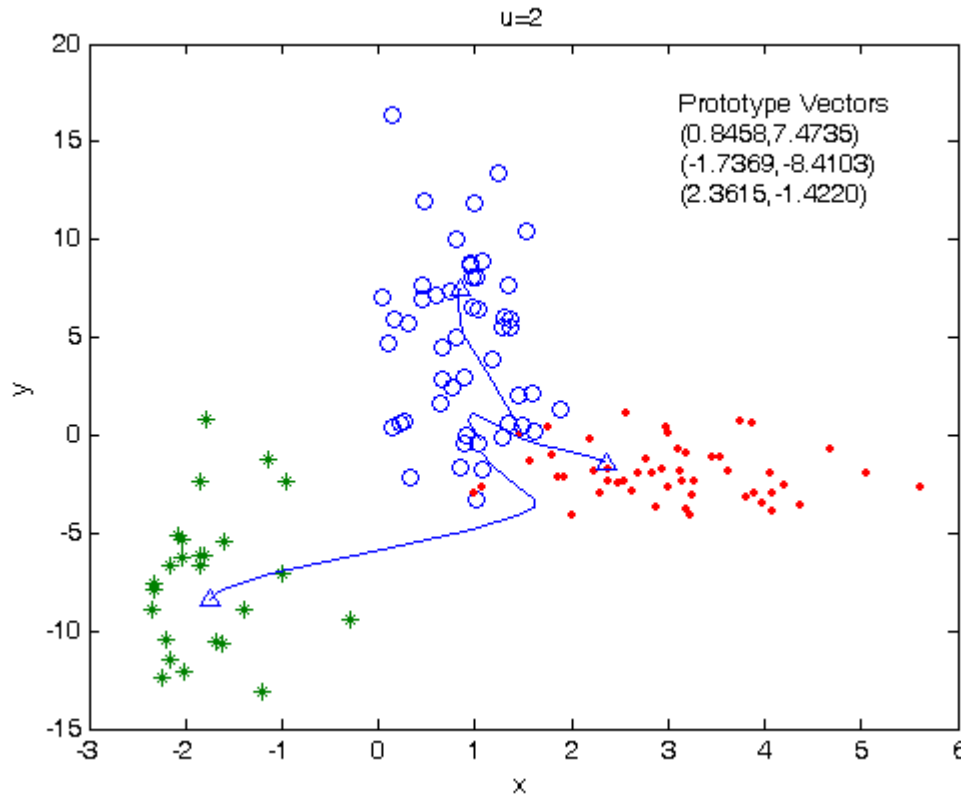


Figure 19: Data vectors (circles, stars, dots) and prototype vectors (triangles) for a three-cluster problem in 2D space.

In the example of Figure 20 the input image (left) was randomly sampled and two-thousand vectors in the RGB space corresponding to the sampled pixels were used as data points. The input data was then clustered using parameter values $Q=2$, $u=2$, $T=0.001$. The algorithm converged after twenty-one iterations resulting in prototype vectors $Y_1=(0.3282, 0.3408, 0.1221)$ and $Y_2=(0.6491, 0.6928, 0.2338)$. The input image was then filtered and Class-one and Class-two images are shown in Figure 14. Class-one (two) images are obtained by preserving all pixels that are closer to Y_1 (Y_2) and setting all other pixels white.



Figure 20: Original image (left), Class-one pixels (middle), Class-two pixels (right).

The algorithm was then applied to the same input data points as the previous example using three clusters. The resulting prototypes are $Y_1=(0.7264, 0.7774, 0.2866)$, $Y_2=(0.2792, 0.2958, 0.1055)$, $Y_3=(0.4815, 0.4877, 0.1697)$.

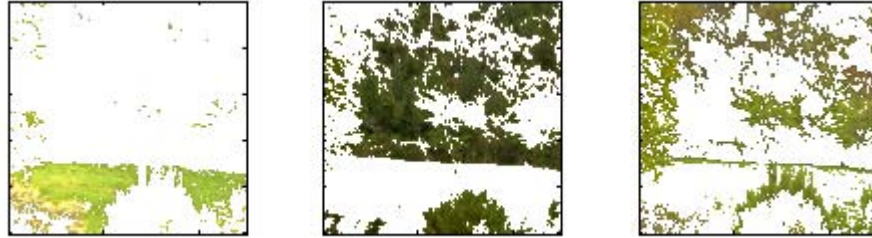


Figure 21: Class-one (left), Class-two (middle), and Class-three pixels (right) of the image.

In the example of Figure 22 randomly extracted pixels from the top-left image were used to compute two prototypes based on two-class clustering. The input image was then processed and the top-right images show the results. The process was repeated using three-class clustering, and the bottom three images show the respective filtered images.

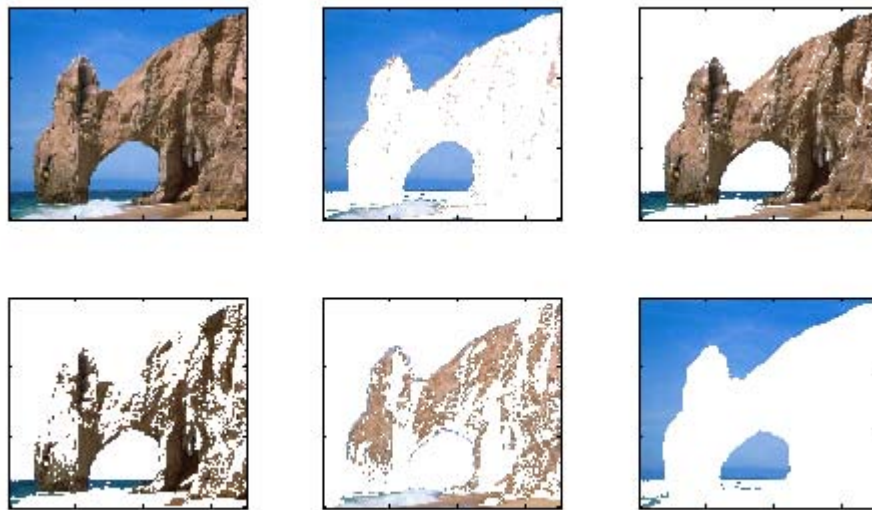


Figure 22: Input image (top left) is processed by filters based on two-cluster (top) and three-cluster (bottom) classification.

In the following examples the input image is partitioned in a hierarchical manner. First, the image is sampled randomly and samples are clustered into two classes using the fuzzy clustering algorithm described above. This leads to computation of two class prototypes.

The input image is then partitioned into two classes in accordance to pixel-prototype distance. This leads to two images, each comprised of input image pixels that belong to the respective class with all other pixels set white. Each of the two generated images is then treated as a new input image and is partitioned into two classes, resulting in four new images. The process continues for a user specified number of partition rounds. Number of generated images generated in each round is double the number of images generated in the previous round.

The image of Figure 23a contains three classes, namely background, leaf, and bug. In the first simulation, the input image was randomly sampled and the selected pixels were partitioned into three classes using fuzzy clustering. The input image was then segmented in accordance to pixel proximity to three prototype pixels. The result is shown in Figure 23b.

In the simulation of Figure 23c, pixels were selected randomly from the input image of Figure 23a, and were partitioned into two classes in the same manner as before. Images of Figure 23c show the result of this two-class segmentation. The Class-one image (leaf and bug only) was then sampled randomly, and selected pixels were clustered using fuzzy clustering, resulting in computation of two prototypes. The image (Class-one) was then partitioned using the computed prototypes. The image of Figure 23d shows the segmentations result.



Figure 23a: Input image

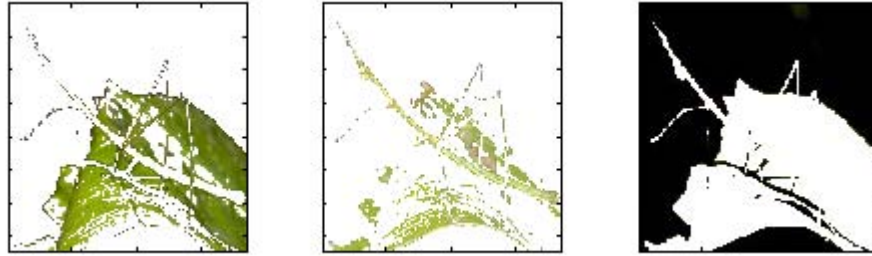


Figure 23b Input image is partitioned into three classes directly

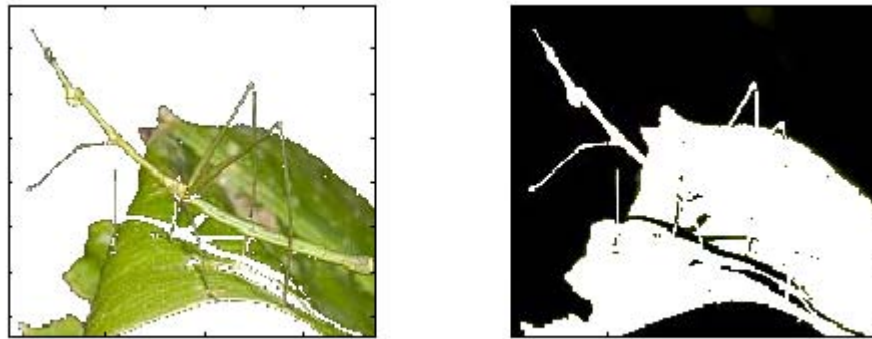


Figure 23c Input image is partitioned into two classes

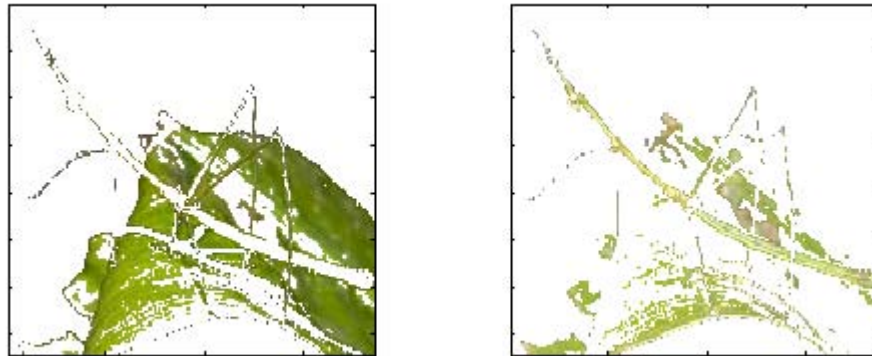


Figure 23d Image of leaf and bug (left image in Figure 23c) is partitioned into two classes

In the following example the image of Figure 24a was partitioned into four classes directly and the result is shown in Figure 24b. In the next simulation we partitioned the original image into two classes and each of the segmented images was further partitioned into two classes. Images of Figure 24c illustrate the result of this operation.



Figure 24a: Input image

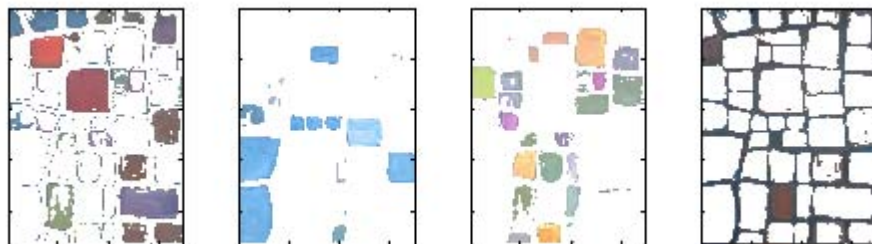


Figure 24b Input image is directly partitioned into four classes

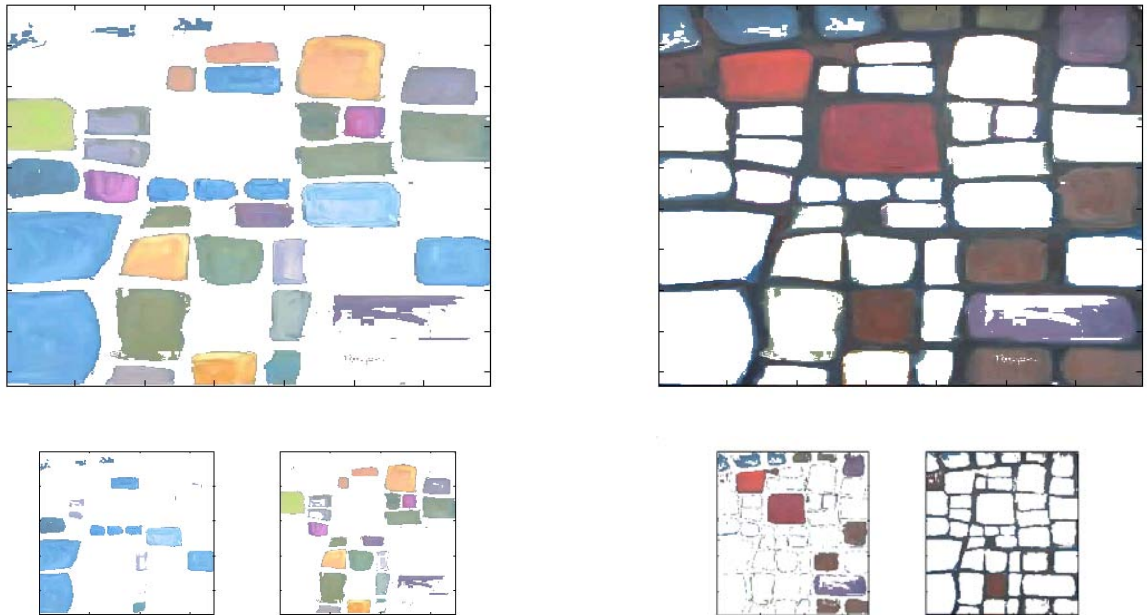


Figure 24c Input image is partitioned into two classes (top) and each generated image is further partitioned into two classes (bottom)

Bayesian Network Modeling

Introduction

Bayesian analysis is a very powerful statistical modeling tool with potential applicability to a wide-range of fields. Virtually any system can be statistically modeled with a so-called Bayesian Network (BN). With a number of limitations, a Bayesian Network can capture causal, statistical, and temporal relationships. Bayesian statistical inference methods have revolutionized the field of practical data analysis in recent years and have acquired the most prominent role in cutting edge research in fields as diverse as VLSI and communication networks. In Bayesian statistics observations from experimental data are combined with probability distributions, based on accepted models of the system, for all quantities of interest.

This section reviews and investigates the requirements necessary to construct models of military system(s) and/or operations, and apply Bayesian methodologies to those models to obtain pertinent information regarding potential failure and performance as a function of the current state of the system(s), and relevant archived experiential data of the system(s).

To be of utility, BN models need to very accurately reflect the character of the target system(s). This creates a need to capture complex, often ambiguous causal relationships between various entities in the system. Therefore, accurate simulation of the behavior between the entities requires the solution of large-scale models which interact at various levels of abstraction. Furthermore, BN models of this magnitude are computationally intractable, requiring $O(n^n)$ time for computation, where n is the number of entities in the simulation. This situation forces simulations to be grossly oversimplified to enable a computationally rigorous solution in a reasonable amount of time, or it forces relaxation of the degree of accuracy of the models by utilizing heuristic (approximate) algorithms, which by their nature can't guarantee optimal solution, and hence yield suspect results. This quandary is further exacerbated by the likelihood that much of the data needed for solution may be inadequate (missing), of dubious validity, and possibly separated by temporal and/or spatial locality depending on the nature of the modeled system.

Thus, this is not just a problem which can be solved by application of a clever statistics algorithm, but requires a simulation framework that is much larger in scope. The development of useful techniques for modeling these systems necessitates that each of the aforementioned problems be addressed. Figure 25 illustrates the complexity of the interrelationship between various facets of the problem. To be effective, a prognostic and diagnostic modeling environment must encapsulate these relationships at a minimum.

Effective Prognostic and Diagnostic Modeling Environment

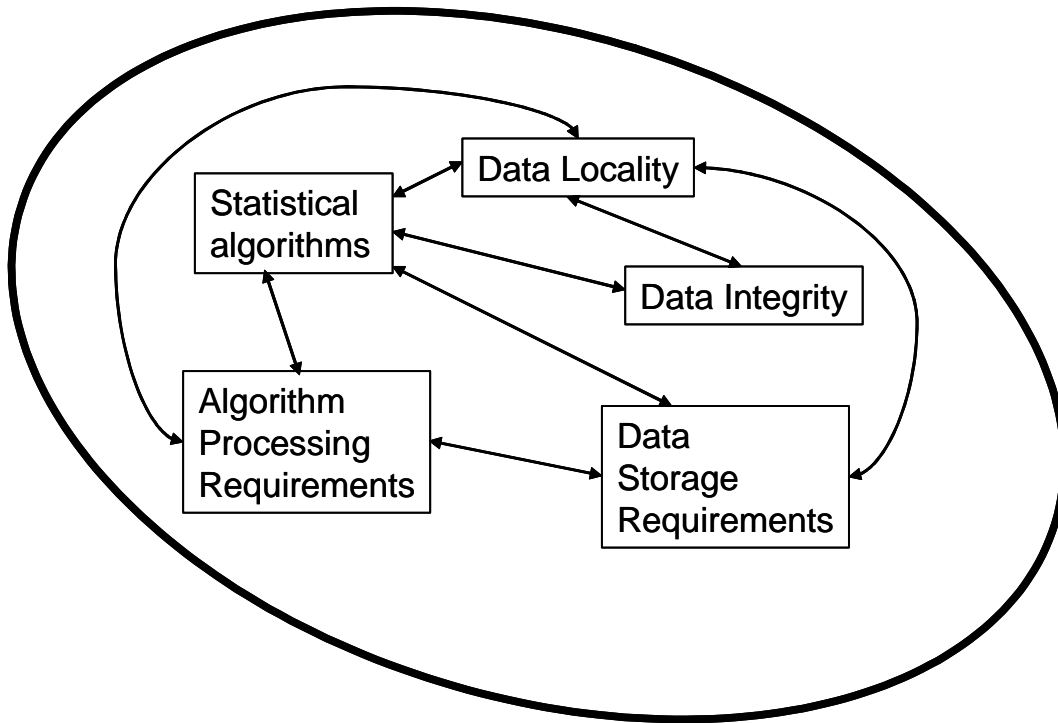


Figure 25: Effective Prognostic and Diagnostic Modeling Environment

In addition to the desired qualities illustrated above, an effective modeling environment must be highly scalable, capable of the simulation of a wide-range of problem size and complexity. For instance, the types of systems to be modeled may range from small (tens of components), to medium (thousands of components) to large (millions of components).

For example, a small system may represent a single circuit board within a weapons system or it may represent the hydraulic system of an armored vehicle. A circuit card has a small number of components (e.g. resistor, capacitor ...) that are coupled together by the circuit architecture, and the hydraulic system has a small number of mechanical parts (e.g. actuators, pumps ...). Test points within the circuit can be monitored to supply ongoing data and monitor the circuit's performance, and sensors (e.g. pressure, position...) can monitor the status of the hydraulic system. The components, their interaction and their state, both current and past, can be modeled to provide a prognostic capability to those systems.

Similarly, a medium system may be a radar system, or an entire vehicle, which can be represented by a collection of small systems, with an aggregate component total in the hundreds or thousands. For example, a radar system model consists of a collection of many subsystem models (e.g. Circuit1, Circuit2, antennae array...) while the armored

vehicle system model is constructed of its various sub-system models (e.g. hydraulics, drive train, electrical...). The interaction between the sub-systems characterizes the required data flow paths, and processing requirements of the composite models.

At the extreme, it may be desirous to model an ongoing military operation that consists of a large number of medium sized systems (e.g. tanks, radar systems, helicopters, ground troops...). This may represent the interaction between hundreds of thousands, or possibly millions of different entities. At this level, the efficient interaction of these sub-systems and effective orchestration of the overall system will determine the outcome of the operation. The performance level and/or failure rates of individual entities within the operation will have an effect on the entire system. Assuming that all the relevant data and algorithms can be properly organized and applied appropriately, and given sufficient computing resources, it is conceivable that Bayesian prognostic techniques can be applied to these types of large-scale systems.

It is useful to approach the overall problem with these types of abstractions in mind. Obviously, with only a few entities, the solution requirements are trivial, but as problem size grows, the computational resources required to effect solutions become significant. Herein lies the crux of the problem. The solution of complex, high fidelity models requires direct confrontation and cognizance of the issues associated with efficient management and utilization of computational resources. This effort establishes guidelines with which to structure a generic modeling framework that is scalable, and spans data type, data acquisition methodology, data location, algorithm choice, and processing capability.

Bayesian Networks and Decision Graphs for Complex Systems

Unlike the simplistic Naïve Bayesian method, very complicated decision trees can be built with Bayesian Networks that allow for communication paths in many different directions. These decision trees are also known as directed acyclic graphs (DAGs) and an example is shown in Figure 26. Communication paths in DAGs can be blocked if information on certain variables is known. This blocking causes the variables to be “separated” and hence Bayesian scientists use the term “d-separation”.

The DAG illustrated in Figure 26 shows several interesting features. The first is serial connectivity. A serial connection is one where communication can pass from one variable to the next along a serial line. An example from the DAG in Figure 26 would be the $A \rightarrow C \rightarrow E \rightarrow F \rightarrow G$ connection. If any information on C, E, or F is known (i.e. instantiated) then communication between A and G will be blocked. Hence A is d-separated from G. In other words, knowing something about C, E or F is enough to make G independent of A. A more precise definition of d-separation is, “Two variables A and B are d-separated if for all paths A and B there is an intermediate variable V such that either (1) the connection is serial or diverging and the state of V is known or (2) the connection is converging and neither V nor any of V’s descendants have received evidence. If they are not d-separated, they must be d-connected.” (Baldwin, 2002)

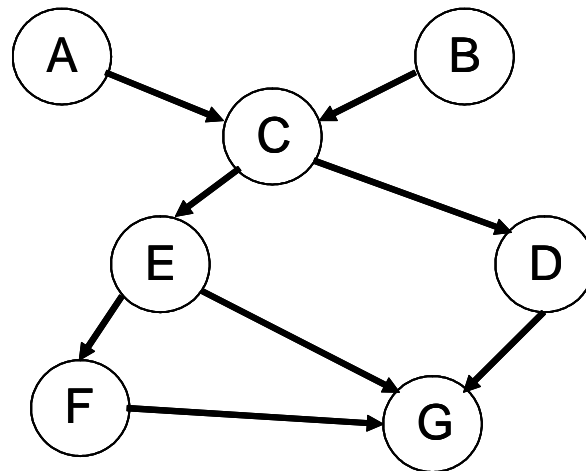


Figure 26: A directed acyclic graph (DAG).

The second interesting feature is the diverging connection. We have already discussed this in some detail in the Naive Bayesian section. In this case, if the parent is instantiated, the children are d-separated and hence become independent. An example of a diverging connection would be the $C \rightarrow E$ and $C \rightarrow D$ connection. Here D and E are children of C. If we know the value of the C variable, D and E become independent.

The final feature is the converging connection and is quite different from the diverging connection. An example of this type of connectivity is demonstrated by the $E \rightarrow G$ and $D \rightarrow G$ connectivity. Parents of G, i.e. D and E, are independent if G is not instantiated. In this case, new evidence or information about G, forces D and E to become dependent.

Pictured below in is an example (Baldwin, 2002) of how to apply d-separation information in DAGs to understand Bayesian analysis. Figure 27 illustrates a DAG that has C, D, G and H having evidence indicating that they have received information. In this case, although all neighbors of E are instantiated, E is still d-connected to F, B, and A because of the rules set forth previously for diverging and converging connection. Therefore, one can determine that the variable E can still affect the variables A, B and F even though we know something about E's neighbors. This plays a very important role in determining the probabilities of A, B, and F and their effects on E even though the events in C, D, G and H are already known.

The strength of this technique is clear. One can change connectivity to apply to a specific problem in the field and can easily change the algorithm accordingly. Also, additional variables can be inserted and one only needs to know its connectivity to the other variables to evaluate the new set of rules to determine the d-separation. Once applied, the algorithm can be altered to include the new rule.

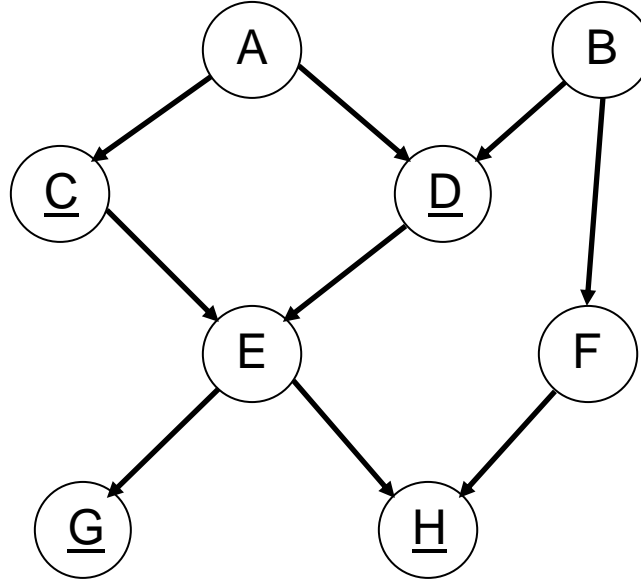


Figure 27: A DAG with C, D, G and H having evidence.

In order to get a better understanding of this technique, let's apply the mathematics to such a diagram (Looney, et.al. 2003). Take for example the simple DAG in Figure 28. Assuming that information about each of the variables is known and illustrated in terms of Truth tables. Example tables are shown in Figure 28. The tables give us information about the states of the variables. For example, the $P(C=F_c | E=T_e)$ is 0.3. In other words, the probability that C is false given that E is true is 0.3. We can use this information to then calculate any probability for any combination of variables using the well known equation:

$$P(A,B,C,D,E) = P(A)P(B)P(E/A,B)P(C/E)P(D/E) \quad \text{Eq. 22}$$

For example, we can determine that the probability of A, D and E being true and B and C being false is simply:

$$P(T_a F_b F_c T_d T_e) = P(T_a)P(F_b)P(T_e/T_a, F_b)P(F_c/T_e)P(T_d/T_e) \quad \text{Eq. 23}$$

Inserting the numbers from the Truth tables and we see that

$$P(T_a F_b F_c T_d T_e) = (0.4)(0.2)(0.4)(0.3)(0.2) = 0.000192 \quad \text{Eq. 24}$$

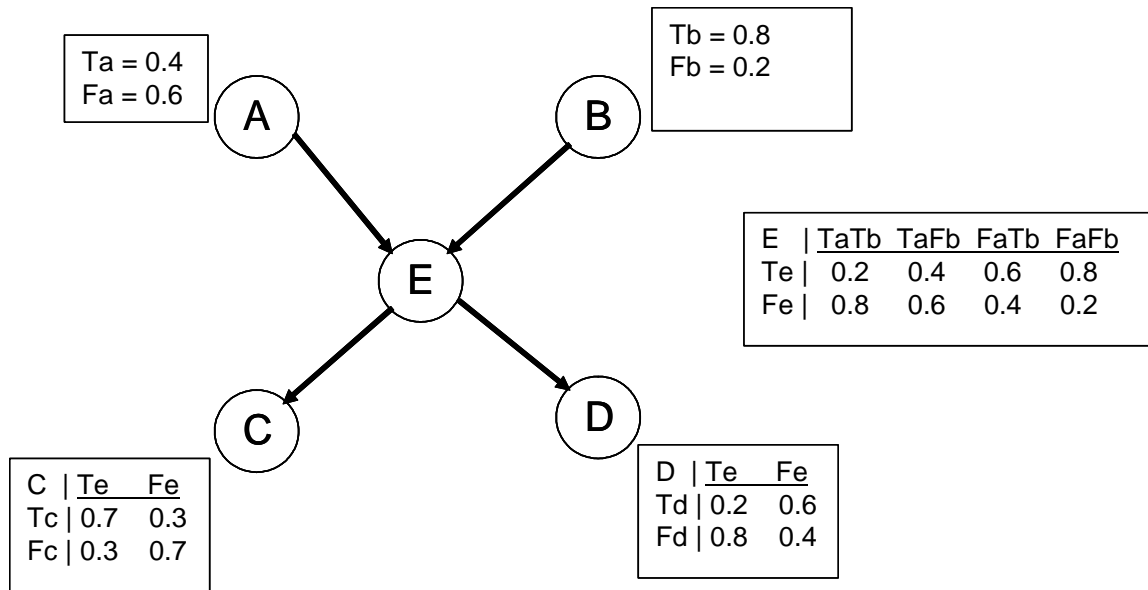


Figure 28: A simple DAG illustrating Bayesian mathematics.

This section reviewed examples of calculating Bayesian statistical inference based on various graph scenarios. As illustrated, the primary methodology for representing a Bayesian system or network is the directed acyclic graph (DAG) data structure. Any non-trivial DAG node can be classified as either divergent, convergent, or a combination of the two. Repetitive application of these methods provides a means of traversing the graph, and updating the probability tables associated with each node. The DAGs utilized for the examples represent a generic system with causal relations established between each of the nodes.

Although understanding the methodologies used to calculate Bayesian statistics is necessary, it is certainly not sufficient information for developing effective system simulations for prognostics and diagnostics. There are at least two other non-trivial issues which must be considered:

- Transformation of a real-world system into a DAG data structure, and identifying the causal relationships between entities (nodes) within the structure.
- Mapping the data in a derived DAG for solution in a distributed environment.

These issues will be examined in the following sections.

Modeling Framework

This section reviews and expands on a methodology developed by the investigators for constructing a computational framework for Bayesian Network applications. This includes derivation of appropriate DAG structures for real-world systems and creation of a generic specification of the requisite data structures for computational implementation. Successful encapsulation of the character of a physical system can be accomplished by converting its components into nodes, and the interaction of the components into

reliability tables, and directed actions. This requires an analytical, systematic approach which can be utilized at different levels of abstraction.

There are a number of techniques utilized for modeling systems and data. Decades of computer modeling and simulation have led to a number of competing and complementary methodologies, which are traditionally lumped under the common headings of Software Engineering or Systems Engineering. Most of these methods can be lumped into two basic categories; 1) Structured Analysis (Functional analysis) and 2) Object Oriented analysis. Structured analysis tends toward description and modeling of the function of systems. This leads to a simulation's development in terms of processes. The processes are typically used to specify direct transformation of specific inputs to outputs, and sub-functions are derived from higher level functions. Functions are then grouped together based on successive steps in the execution of higher level function. Object-oriented analysis is based on the interaction of actual entities or defined components. The functions are grouped based on their interaction with a specific data type or entity.

While both types of analyses have their pros and cons, object-oriented analysis is particularly well suited to deriving DAG representations of systems since it naturally defines objects based on physical, real-world constraints. As opposed to Structured Analysis that is more concerned with the process of interaction, often at the expense of data fragmentation. (Balin, 1989) describes a methodology which utilizes a hybrid of Entity Relationship Modeling (ERM), and Entity Data Flow Diagrams (EDFD) to describe objects and their interactions. ERMs are utilized to establish entities, and generalize interactions. EDFDs are created from the ERMs and identify the relationships more specifically. These methods can be recursively employed to yield hierarchical entity relationships, which can be easily represented by DAGs.

For instance, consider an active helicopter, which it is desired to model. For purposes of discussion, a simple version of the top-level interaction is presented below in Figure 29. Entities are represented by the rectangles, in this case; PILOT, ELECTRONICS, MACHINERY, WEAPONS and ENEMY. Interactions between entities are denoted with the diamond shapes; FLIES, FIRES, INFORMS, MONITORS, AIMS, ACQUIRES and DESTROYS. Note that no direction of action has been assigned, although some relationships are apparent (e.g. the pilot flies the machinery), in general the nature of cause and effect are not always so apparent. At this stage, it is left ambiguous. Note that each entity has either an 'A' (active) or a 'P' (passive) label associated with it. An entity is active if it operates on inputs to produce outputs. A passive entity takes no action of its own, but is acted upon. An active entity may receive actions as well as producing actions. In this case, PILOT, ELECTRONICS, and WEAPONS are denoted as active entities while ENEMY and MACHINERY are denoted as passive entities. This diagram can easily be enlarged to take on a larger scope. As an example, ENEMY could have his own weapons (e.g. ENEMY_WEAPONS) in which case, ENEMY would become a passive entity as well. Or, another helicopter could be added to the situation, their ELECTRONICS could interact, as could their individual PILOTS, and they could both target the same ENEMY entity.

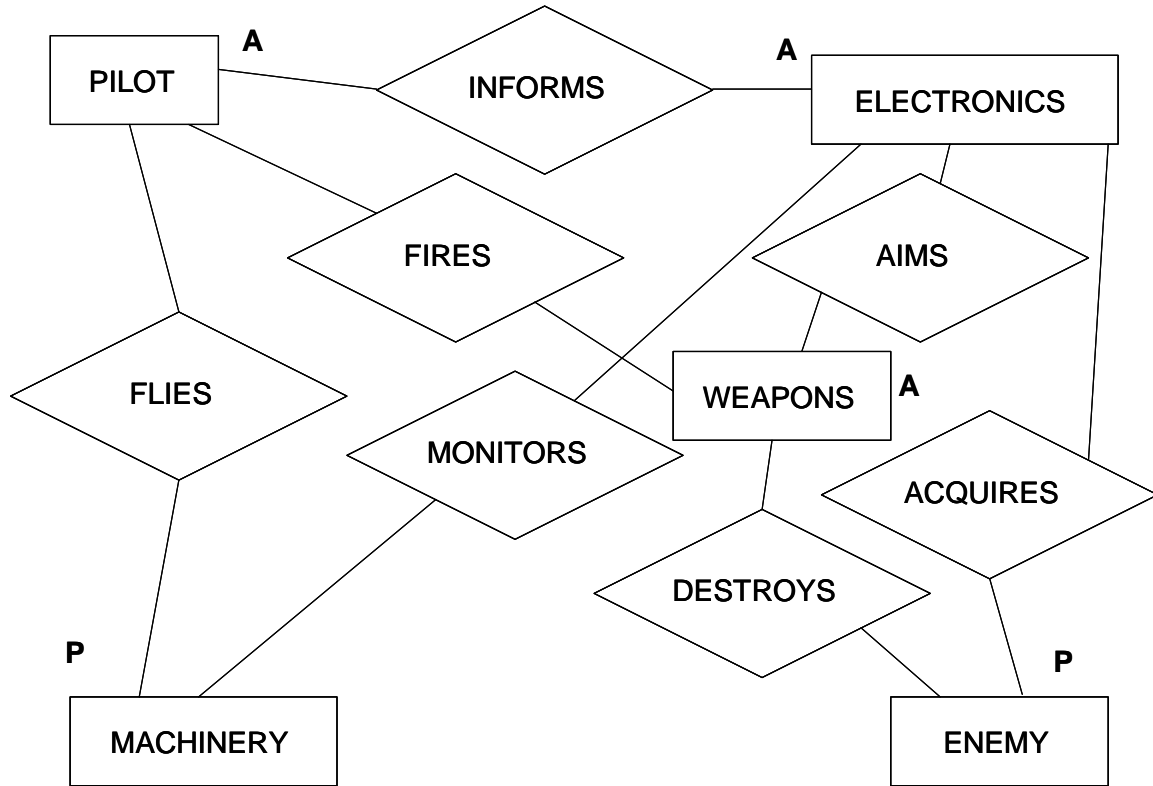


Figure 29: Helicopter, Level 0 ERM

A corresponding top level (Level 0) EDFD is illustrated in Figure 30. This diagram shows the direction of data flow (interaction) between the top level active entities from the Level 0 ERM. All the entities that were previously defined in the ERM are assigned numbers and accounted for in the EDFD and enclosed in brackets, but the interactions are modified. The active entities are denoted by circles, while the passive entities stand alone and typically will represent an existing database, or a real-time stream of data (e.g. sensor data from the machinery, or position information on the enemy).

Note that the diagram is looking more like the DAGs that are required for Bayesian network simulations. This diagram can be manipulated into a high-level DAG despite the fact that cycles exist in the figure, as is illustrated in Figure 31. The cycle PILOT → MACHINERY → ELECTRONICS → PILOT that exists in the EDFD is permissible in this instance because the entities are at the same level (share the same parent) in the DAG. This is referred to as “crosstalk” between first generation children of a node, and is represented by the dot notation between the nodes. As discussed above, this communication between children is valid unless the parent node has been instantiated, then they are “d-separated”. Crosstalk is not, however, permitted between nodes at different levels within the DAG structure.

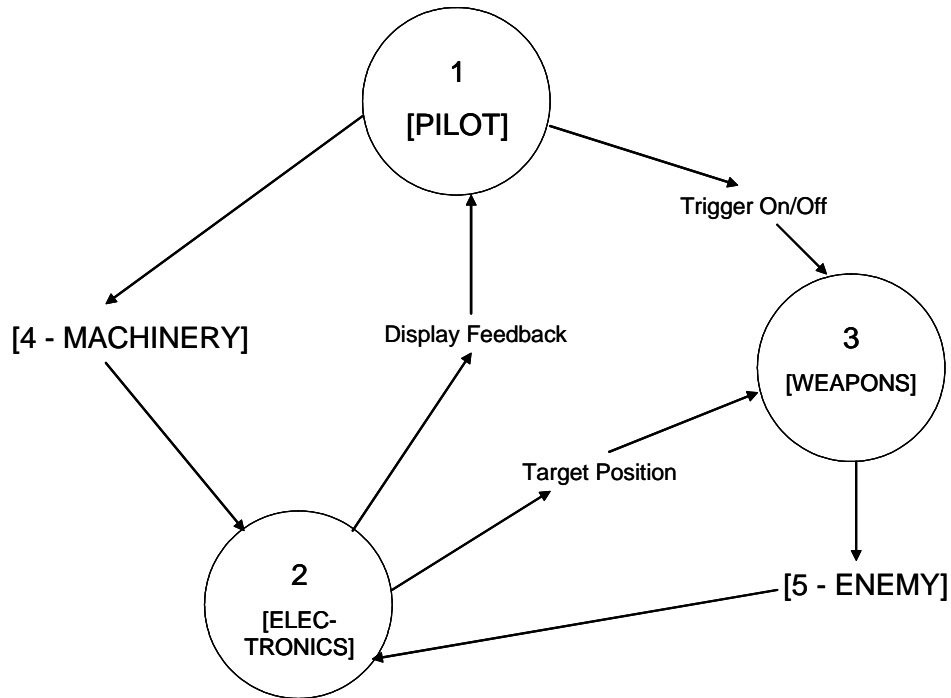


Figure 30: Level 0 EDFD

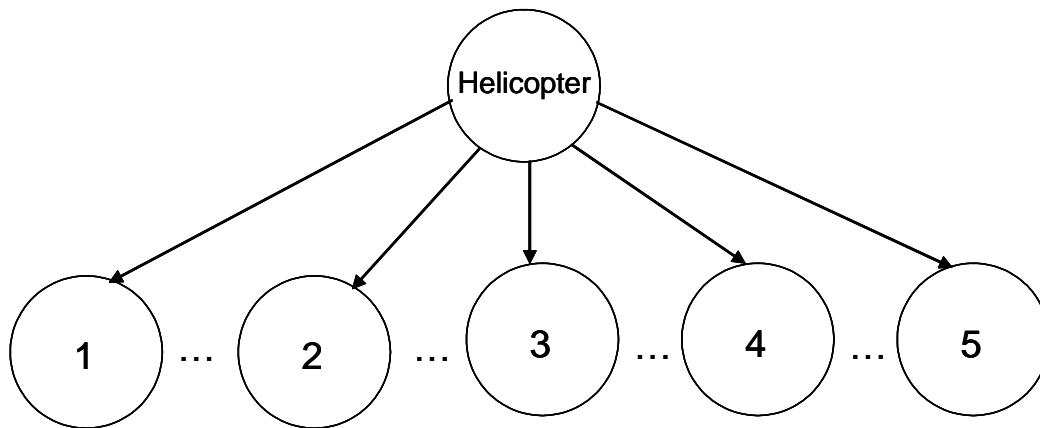


Figure 31: DAG Bayesian Network from Level 0 EDFD

The top-level EDFD can now be recursively decomposed into sub-entities and/or functions as illustrated in

Figure 32. In this case, entity 2.0 – ELECTRONICS has been decomposed into eight sub-entities. Entities 2.1 – 2.5 represent new Active functional sub-entities, and entities 2.6 – 2.8 represent new Passive entities (typically data). As above, Active Entities are denoted in the circles. Note that as in Figure 30, some of the entries in the diagram are not entities. This denotes intermediary data between active entities. For instance,

“Aircraft State”, “Aircraft Position” and “Weapon Status” are inputs to entity 2.1 [OUTPUT DATA]. Any of this data could be promoted to a passive entity status, but in this case, since entity 2.6 DISPLAYED DATA represents a conglomeration of the input data to 2.1, it would simply be redundant. The interaction of 2.0 ELECTRONICS with the active entities 1.0 PILOT and 3.0 WEAPONS is denoted with the circles 1 and 3, and the interaction with passive entities 4.0 MACHINERY and 5.0 ENEMY is denoted with [4] and [5] in the figure.

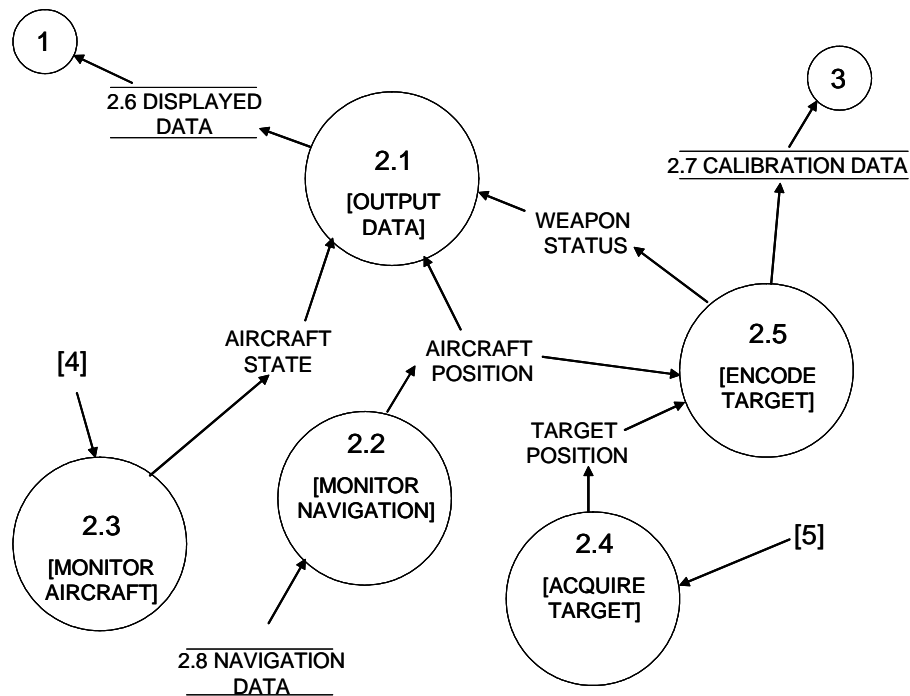


Figure 32: Level 1 EDFD for Electronics (2.0)

Figure 33 is a composite DAG that represents the relationships derived in both Figure 30 and

Figure 32. Although not illustrated, cross-talk between children at each level in the acyclic graph, or “tree” structure is permitted according to the rules of “d-separation”.

A DAG for the entire HELICOPTER can be constructed in this manner by recursively decomposing each active entity and sub-entity in the EDFD. ERM can be introduced at any point where the relationship between entities is difficult to discern. Fortunately, the properties of “d-separation” provide some allowance for resolution of ambiguous relationships. The entities can be decomposed to any level of desired granularity down to unit component level. The final product will be a large-scale DAG, often referred to as a “hierarchical tree” structure. Efficient representation and solution of this data structure is required for simulation and determination of Bayesian statistical relationships.

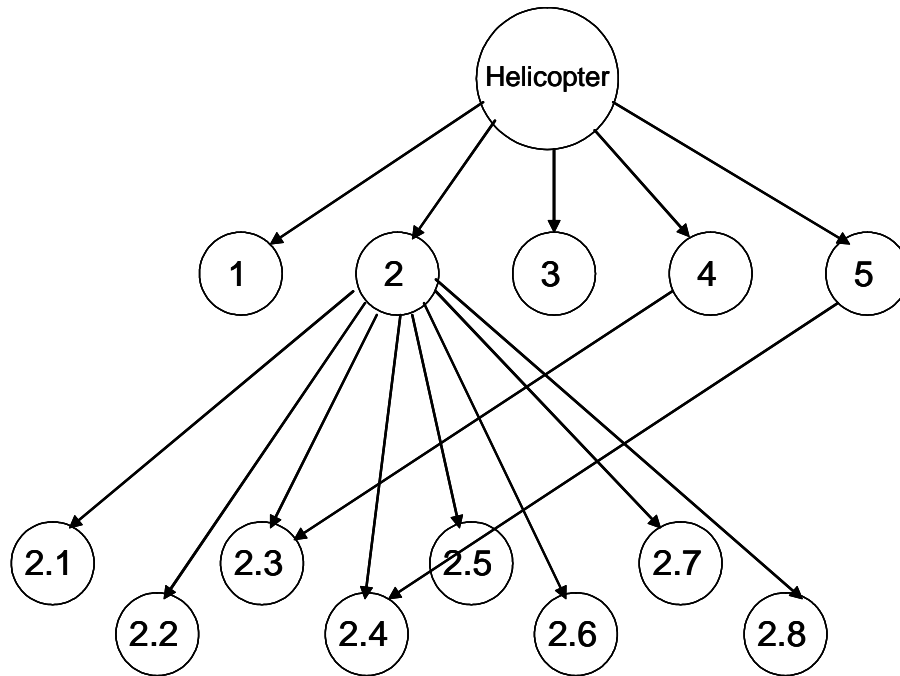


Figure 33: DAG with Level 1 EDFD for ELECTRONICS (2)

Development of practical algorithms requires the introduction of pertinent and efficient data structures. In general, the derived DAGs can be represented as a graph $G(A)$, which is defined by

$$G(A) = (V(A), E(A)) \quad \text{Eq. 25}$$

with the set $V(A)$ of n -elements, called vertices or nodes, (one per variable in a vector x) in the graph $G(A)$, and a set $E(A)$ of unordered distinct vertex pairs called edges which represent connections, or dependencies, between the vertices (variables in x) in the graph $G(A)$. This definition can be associated with a matrix A , where

$$V(A) = \{v_1, v_2, \dots, v_n\} \quad \text{Eq. 26}$$

is the set of all vertices corresponding to the rows of A , and

$$E(A) = \{(v_1, v_2), \dots\} \quad \text{Eq. 27}$$

is the set of all edges within the graph $G(A)$ such that $(v_i, v_j) \in E(A)$ if and only if $a_{ij} > 0$ and $a_{ji} > 0$, where a_{ij} and a_{ji} are the elements in the i th row and the j th column and the j th row and i th column of a traditional connectivity matrix respectively (Jess and Kees 1982). In the case of relatively sparse connectivity, a more efficient directed two-dimensional adjacency list is constructed from the above graph description and illustrated in Figure 34. Consider the DAG model of Figure 33, it consists of fourteen vertices; H (helicopter), the level 0 entities 1 – 5, and the level 1 entities 2.1 – 2.8. Each of the

vertices is represented in the left vertical list. Then, the edges associated with a given vertex are represented by the horizontal list attached to that vertex. An entry in the horizontal list indicates that the vertex has a connection (edge) to each vertex in the horizontal list. Underlined entries indicate that the edge is reversed, while non-underlined entries notate a forward edge connection. This is useful for updating the BN, and providing paths for crosstalk between child nodes.

Linked lists connecting the graph vertices (nodes) in the vertical direction facilitate their removal for inclusion in sub-graphs for recursive partitioning. They can be easily removed from an adjacency list. If an indexed array were used in the vertical direction, the need for pointers to next elements would be required anyway. The need for a List representation of connecting edges (horizontal entries) between vertices is apparent due to the inherently sparse nature of the BN models. A connectivity matrix requires $O(n^2)$ memory storage, vs. $O(cn)$ storage for the List representation. The constant, c , represents the maximum number of adjacent nodes in a BN graph representation. Since the graph is directed, the edges between adjacent nodes, or vertices, complement each other. Taking advantage of this property by using pointers between complementing edges facilitates quick removal of the edges from their respective lists when the graph is partitioned as discussed below.

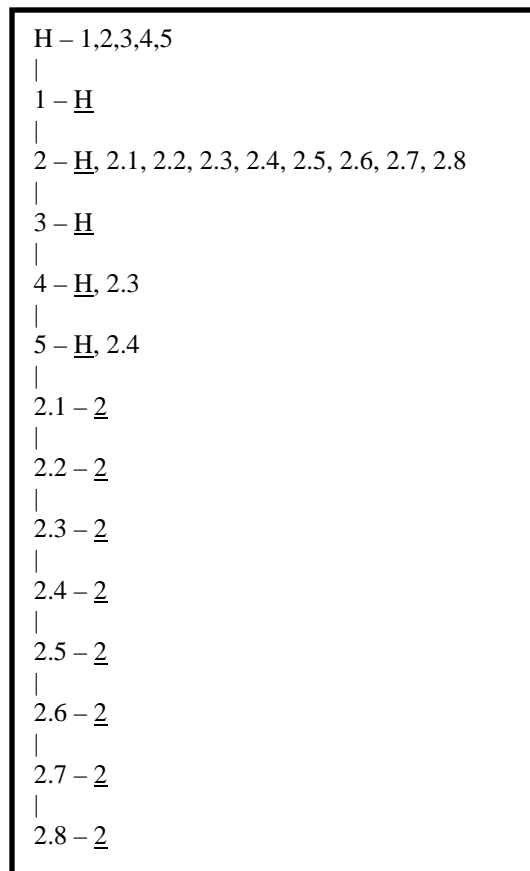


Figure 34: 2-D Adjacency List DAG Representation

Furthermore, graph structures enable $O(n)$ breadth-first and depth-first traversals. This is a useful characteristic. Many algorithms make liberal use of this feature. A typical graph structure specification, dubbed BN_Graph, (Bayesian Network Graph) is detailed in Figure 35.

```
*BN_Graph; //pointer to the graph structure of the BN discretizations

BN_Graph:: //Undirected 2-D Adjacency Graph Structure
{
    Data::
        Vertex_total:: // Integer, Total vertices in graph
        Graph_Effort:: // Integer, Approximated computational effort of graph
        First_Vertex:: // Pointer to first vertex (element) in list

    Methods::
        .....
        .....
};
```

Figure 35: BN_Graph Structure Specification

Figure 36 is a specification of the pertinent members of a vertex from a BN_Graph structure. Consider a vertex_ptr, v. The data member v.weight is a measure of the computational effort required of the given node. This parameter wouldn't be necessary to specify if all entities data were of equal weight. Since there are a variety of node types and corresponding sizes of statistics tables (denoted as BN_table), this parameter must be considered.

```
*Vertex_ptr; // pointer to a BN_Graph vertex

Vertex:: //Data Structure describing a node in a BN discretization,
        BN_Graph member
{
    Data::
        Element_ID:: //Integer, Global Entity Number
        Weight:: //Integer, Computational weight of individual vertex (element)
        Next_vertex:: //Vertex_ptr, next vertex in list
        Prev_vertex:: // Vertex_ptr, previous vertex in list
        Edge_list:: // List of connecting edges to adjacent vertices (elements)
        BN_table;; // L

    Methods::
        .....
        .....
};
```

Figure 36: BN_Graph_Vertex Specification

The preceding discussion illustrates a straight-forward methodology to convert a system into a DAG, then a data structure suitable for implementation in a computer code. As noted, these system BN_Graphs can range in size from a few to possibly millions of nodes. The next consideration is how to effect the partitioning of large-scale BN Graph assemblages to enable solution in a distributed environment.

BN – Graph Partitioning for Simulation in a Distributed Environment

Viewed as a global objective (i.e. without partitioning of some sort) the maximum problem size quickly reaches its peak as the global assemblage exceeds the allowable space and computational capability. Therefore, some type of graph partitioning or problem division must necessarily be called upon as problem size grows. The question is not if to partition, but where, when, and how to apply partitioning.

Solution of these very large problems absolutely necessitates the use of domain partitioning. Acceptance of this constraint demands that particular attention be given to the fact that inefficient, haphazard grain extraction and partitioning techniques can significantly increase the required processing time of large problems. This, coupled with the inordinately large storage requirements arising from some techniques, requires that serious consideration be given to these issues to minimize their effects. Obviously, these are not issues with small problems, but they become severely debilitating factors as problem size grows.

Since the determination of an optimal decomposition of a general graph system is a NP-Complete task (Yannakakis 1981), any useful heuristic decomposition algorithm that is capable of running in polynomial time cannot guarantee “optimal” results from a general graph structure.

The product of the decomposition algorithm will be a hierarchical tree, similar to the “e-tree” described by (Jess and Kees 1982) and discussed below. These structures describe precedence relations for solution of a given problem. Techniques for distribution of these trees for processing in a distributed, possibly heterogeneous, computing environment have also been investigated. The goal being to maximize the concurrency found in the hierarchical decompositions.

The advent of widespread parallel and concurrent processing has ushered in a new genre of schemes, which have different goals than simply minimizing the memory requirements of a sparse system. These schemes, variously labeled as “secondary reordering,” “e-trees,” or simply “graph partitioning,” attempt to isolate regions of the DAG structure by minimizing the level of connectivity between adjacent regions of isolation. These approaches seek to maximize the concurrency while minimizing the communications overhead due to boundary areas common to adjacent regions.

A number of researchers have investigated various methods for physically decomposing system graphs. The techniques go by many different names, take many different forms, have many different algorithmic complexities, and produce many different results. This potpourri of techniques can be attributed to the fact that it is NP-Hard to find the maximally balanced connected partition for a general graph (Chlebikova 1996). Thus, a myriad of heuristics exist which attempt to approximate the partition in polynomial time. Graph partitions are referred to by a variety of names including; substructure, sub-domain and sub-graph.

The usual stated goal of graph decomposition or partitioning methods is to maximize the degree of load balancing while minimizing communication costs in a given parallel environment. Load balancing implies that substructures within a given decomposition have approximately the same computational effort. Communication costs are proportional to the size of the interfaces between adjacent partitions, which are the “cut” edges in the system graph. Most of the research, however, has emphasized only the issue of minimizing communication overhead, with little regard to network tuning and balancing.

(Jess and Kees 1982) introduced a model for parallelization dubbed the “elimination-tree” or simply “e-tree”. The e-tree is a spanning tree for the given graph. The critical path in the e-tree indicates the minimum number of computations necessary to complete the BN_Graph computation. The e-tree is presented as a data structure to guide parallel processing and takes advantage of some sophisticated graph theoretic techniques. Simply put, a mapping function assigns a “label”, γ , to each vertex: V in the set $\{1,2,\dots,n\}$. “Label classes” are defined as an ordered set, or list, of labels and contain vertices which can be processed in parallel after the vertices of all previously defined classes have been processed. A symbolic factorization technique is used to create these label classes, or sub-graphs, from $G(A)$. Based on the resultant labeling configuration, an elimination tree is constructed. The e-tree defines the precedence relations that must be strictly followed to obtain a perfect elimination graph. Pivots at the same level within the e-tree indicate that processing can be conducted concurrently. The algorithm for obtaining the e-tree has complexity $O(n^2)$.

(Liu 1987) proposed a tree rotation algorithm that searches for cliques within an e-tree, which can be arbitrarily renumbered without affecting the graph properties. Vertices from each clique are candidates for tree rotation nodes. The goal of these rotations is to minimize the depth of the tree, thereby increasing the level of available concurrency. (Alvarado 1992) describes an $O(n)$ complexity algorithm which examines the vertices of the elimination tree in increasing order, beginning with the leaf nodes. This process seeks to identify nodes that can be independently shifted between levels in the e-tree.

The edges between cut (partitioned) branches of the tree represent communication between different machines in the distributed environment. The “critical path” of the solution, which is equal to the longest path from branch to root in the above tree, is a function of the computational effort (FLOPs) required for the statistical updates of each branch, message size, and network latency. (Note that each branch is processed entirely

on one machine.) It represents the minimum possible elapsed time to complete a probability table update solution of the given BN graph mapping in a distributed environment.

The size of individual messages depends on the size of the data update tables which exists between the two substructures represented by the tree nodes in question. Network latency must also be considered, and is tightly coupled to the message size. However, it is difficult to model with any accuracy since, in any circumstance other than benchmarking, the user faces contention and has no control over network characteristics.

BN Graph Decomposition Algorithms

This section discusses algorithms for decomposition of large-scale Bayesian Network graph structures. There are two primary reasons for decomposing BN-graphs: first, to enable parallelism, which enhances problem solution time; and second to solve large problems. Decomposing a large problem enables solution in a networked environment, whereas the problem in its entirety may be too large to be solved on a single machine. These two goals are interrelated. While their motivations are distinct, the means of achieving them are similar. The basic premise is to dissect a given BN-graph to achieve an efficient and timely solution. The basic requirements of an acceptable BN-graph decomposer:

- it must handle arbitrary graph geometries;
- it should create sub-graphs which allow the overall computational effort to be evenly distributed among processors;
- it must minimize the amount of interface nodes to minimize inter-sub-graph communication costs; and
- it must be of modest polynomial complexity,

This provides a rough outline of algorithm requirements, and helps constrain the solution space. While optimal results for the first three requirements are highly desirable, this must be tempered by the realization that heuristic algorithms are required. This due to the fact that it is of NP-Hard complexity to find the maximally balanced connected partition for a general graph (Chlebikova, 1996). Thus, the fourth basic requirement is cognizant of this and necessitates that the methodologies be heuristic, and can only approximate the optimal partitions in polynomial time.

This requirement is necessary because a basic premise for decomposition is to speed-up problem solution. If the partitioning scheme itself takes an inordinate amount of computational resources, then its very purpose has been defeated. The solution of BN-graphs has a high complexity (up to $O(n^n)$ effort in the extreme). Thus, the decomposition itself should be of lower complexity to prove useful.

The requirements listed above for a suitable partitioning scheme, are analyzed to demarcate a more formal set of constraints for the algorithm detailed here. A heuristic algorithm has been developed which satisfies the above requirements. Its structure is

based upon bisection of the maximum graph diameter. It is fast, $O(n^{5/3})$ and works well on symmetric idealized problems, as well as “real-world” problems. The design is based upon representation of the Bayesian Network as a graph structure, which enables a number of desirable properties for manipulation and dissection of the topology. The parent graph is recursively partitioned into a number of sub-graphs as prescribed by the computing platform characteristics.

Graph Representation of BN Model

Development of efficient algorithms requires the utilization of pertinent and efficient data structures. In this case, a directed two-dimensional adjacency list is used to represent the graph structure of the Bayesian Network. Consider the BN model and its partitions, illustrated in Figure 37. It consists of eight nodes denoted A – H, and their connecting arrows. The partitions consist of whole nodes (nodes aren’t split between partitions) whereas arrows can be separated.

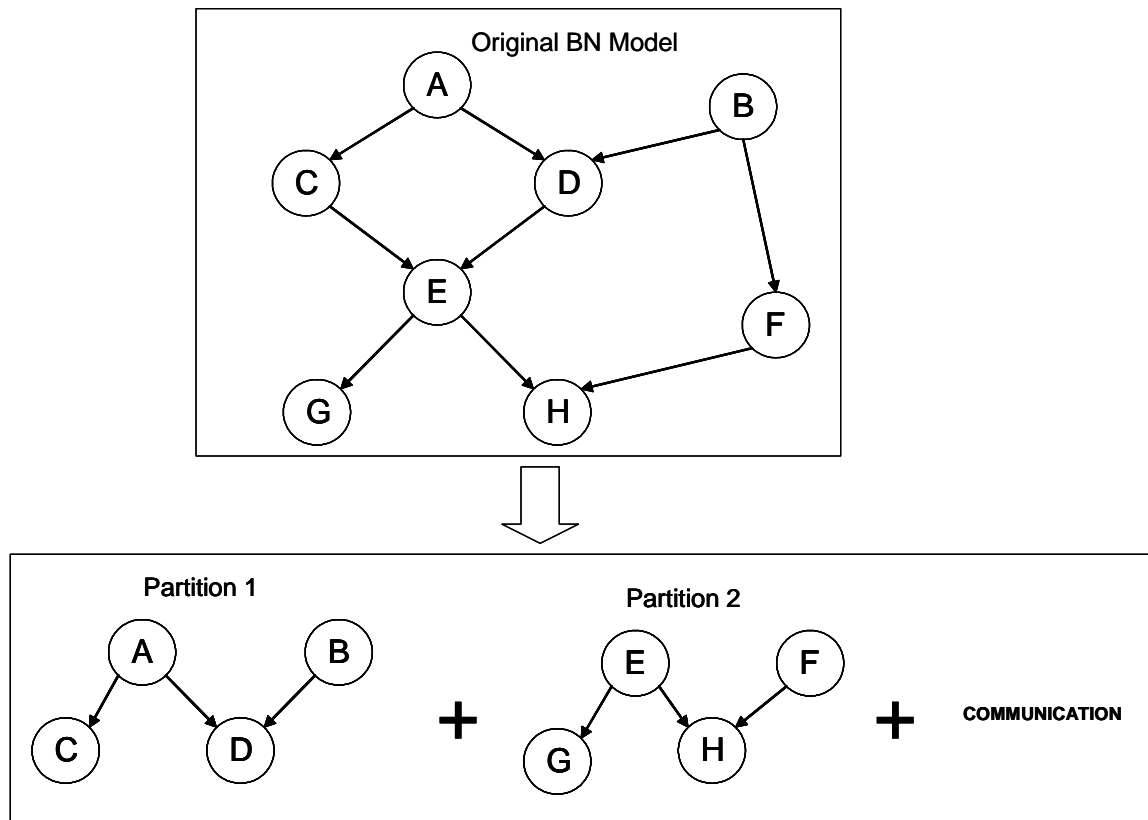


Figure 37: Sample Partitioning of a BN Model

Therefore, the graph representation of the BN models is constructed from the individual vertices, as illustrated in Figure 34. The overlapping of edges is accounted for in the reassembly procedure. The nodes from the BN model are the “vertices” of the graph. The physical connections, or arrows, between nodes are represented by the “edges” of the

graph structure. An underlined node label in the connectivity list denotes a reverse edge in the directed graph, while those not underlined imply a forward edge.

Using 2-D Linked Lists to connect the graph vertices (nodes) in the vertical direction facilitate their removal for inclusion in sub-graphs for recursive partitioning. They can be easily removed from an adjacency list. If an indexed array were used in the vertical direction, the need for pointers to next nodes would be required anyway. The need for a List representation of connecting edges (horizontal entries) between vertices is apparent due to the inherently sparse nature of a typical BN model. As noted above, a connectivity matrix requires $O(n^2)$ memory storage, vs. $O(cn)$ storage for the List representation. The constant, c , represents the maximum number of adjacent nodes in a BN topology. This number will typically be $\ll n$ for a BN system. Hence, the list representation is justified. Since the graph is directed, the edges between adjacent nodes, or vertices, don't mirror each other, so the system matrix is asymmetric, but the reverse edges are carried along to denote communication channels. This construct facilitates quick removal of the edges from their respective lists when the graph is partitioned.

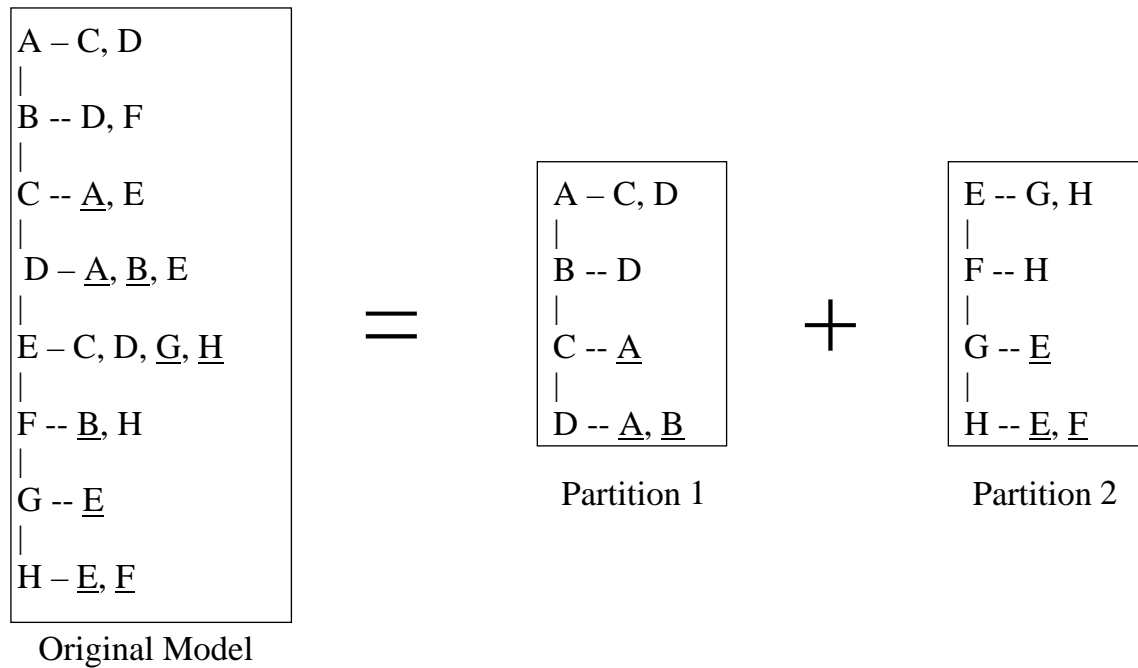


Figure 38: Sample 2-D List Representation of Figure 1.

Furthermore, if reverse edge use is permitted, directed graph structures enable $O(n)$ breadth-first and depth-first traversals. This is a useful characteristic. The BN_PARTITION algorithm makes liberal use of this feature. The graph structure, dubbed BN_Graph, utilized is detailed in Figure 35.

Figure 36 is a specification of the pertinent members of a vertex from a BN_Graph structure. Consider a vertex_ptr, v . The data member $v.weight$ is a function of the

computational requirement to generate the reliability table. It is simply a measure of the computational weight of a given node. This parameter wouldn't be necessary to specify if all nodes were of equal weight. But, since there are a variety of node types, which may be intermingled within a given BN model, this parameter must be considered. For instance, one reliability table may be based on a set of given constant parameters requiring nearly no processing, while another may be created from a large database of test information and experiential data, which requires frequent updating. This provides a means of comparing the relative computational effort of each node. This may be calculated, or obtained experimentally as, $v.weight = \#FLOPs$ where $FLOPs$ is the number of floating point operations required for calculation.

```
*Vertex_ptr; // pointer to a BN_Graph vertex

Vertex:: //Data Structure describing a node in a BN discretization,
        BN_Graph member
{
    Data::
        NODE_ID:: //Integer, Global Node Number
        Weight:: //Integer, Computational weight of individual vertex (FLOPS)
        Reach:: //Integer, Reach from starting vertex (node in a traversal)
        Current_Weight:: //Integer, Weight from starting vertex (node) in a traversal
        Ref_Weight:: //Integer, Weight from reference vertex (node)
        Next_vertex:: //Vertex_ptr, next node in list
        Prev_vertex:: // Vertex_ptr, previous vertex in list
        Edge_list:: // List of connecting edges to adjacent vertices (nodes)

    Methods::
        ....
        ....
};
```

Figure 39: Specification of BN_Graph Vertex

The parameter, $v.reach$, is an indication of the number of nodes or “hops” that v is away from some reference vertex. For example, considering Figure 37, with node “A” as reference, implying $A.reach = 1$, then $C.reach = D.reach = 2$, and $E.reach = B.reach = 3$, finally $F.reach = G.reach = H.reach = 4$. This is each node's respective distance from node “A”. The parameters $v.current_weight$ and $v.ref_weight$ are used to store values obtained by the graph traversals. Specifically, $v.current_weight$ holds the values equal to the node's $v.reach$ multiplied by the node's weight, $v.weight$. This provides a numerical basis for determining the computational cost of including each node with the reference node. The parameter $v.ref_weight$ is used to hold the results of multiple traversals. This is accomplished by summing the contents such that $v.ref_weight = v.ref_weight + v.current_weight$. This enables a quantitative measure of each node's computational distance from more than one reference node. These parameters are utilized to ensure that the graph partitions are balanced.

Communication Considerations

The basic tenant of the partitioning scheme is bisection of the BN graphs maximum diameter at each level in the hierarchy. This methodology is implicitly coupled with the second and third requirements listed above. The second requirement calls for balanced, evenly sized sub-graphs, and the third calls for a minimum interface between sub-graphs to minimize communications. These serve to act as constraints on the diameter bisection methodology.

The number of sub-graphs required to bisection the diameter of the graph is a function of the graph topology. For example, consider the two possible scenarios in Figure 40. Assume each consists of a highly populated uniform 2D-graph with asymptotically large numbers of nodes. Then, both scenarios effectively satisfy the second requirement of balanced, evenly sized sub-graphs. Each sub-graph is equal in size, and the uniform computational effort per node ensures balance.

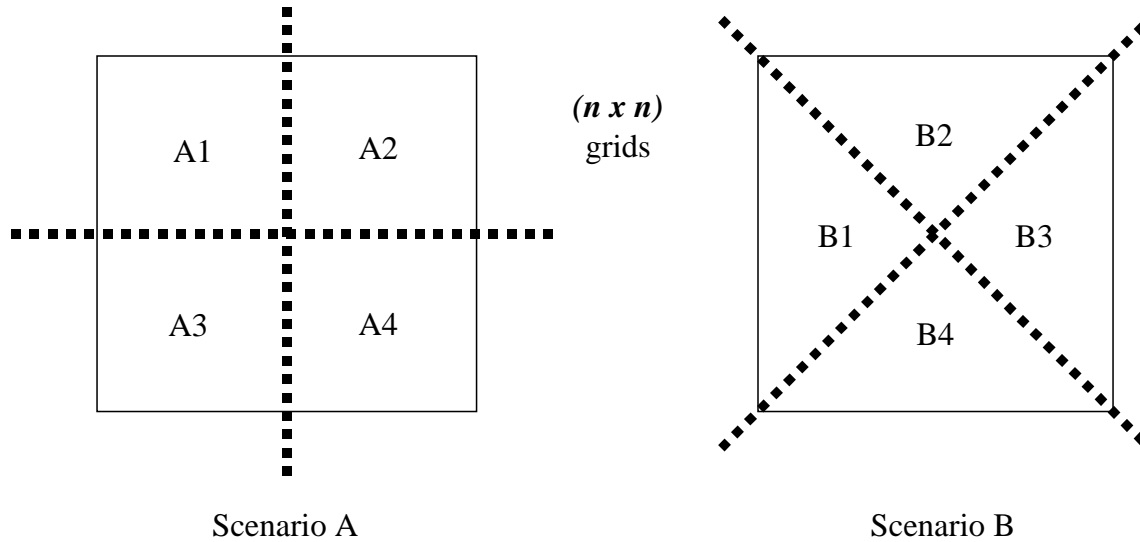


Figure 40: Comparison of Partitioning Techniques

The third requirement calls for minimizing the interfacial zones between sub-graphs, which serves to reduce communication requirements. In this case, the total length of the lines shared between sub-graphs provides a measure of the communication requirements. For scenario A, the length is simply $2n$. However, for scenario B, the length of the interface is $2n\sqrt{2}$. This represents greater than 40% increase in communications cost for scenario B, as compared to A. Clearly, for the third requirement, scenario A is superior.

With regard to the computational effort criteria, both scenarios appear to meet this when viewed in a statistical mean sense. Since the length of a side is given by n vertices, this implies that the diameter of the connectivity matrix prior to partitioning is also n .

For these reasons, Scenario A does the best job of meeting the requirements prescribed. The challenge is to develop an algorithm which meets these characteristics, and meets the remaining two overall requirements as listed earlier. Handling arbitrary topologies as would be encountered in real-world BN problems was the first requirement, and the fourth requirement, which requires a reasonable polynomial complexity for the implementation of the algorithm.

Algorithm Development

Before decomposition can commence, it is necessary to transform the Bayesian Network model information into a graph structure. Typically, the total, variable number of graph nodes is known. Use of the variable n , for nodes is therefore acceptable for determining complexity. Nodes can be catalogued by type. Each node is allocated a vertex structure, $O(n)$, and is inserted into the vertical list as implied by Figure 34. The edges or horizontal lists, which denote the connections between nodes, are determined by assessing shared edges between nodes. Each node is also allocated a structure consisting of a list of all adjacent nodes in which it exists. This list is created as the node information is read from the database. Each node in the adjacent nodes' list inserts the node's I.D. into its own list. After input, each node has a list of all adjacent nodes associated with it.

The edges of the node graph are inserted by going through each connectivity list. Beginning with the first node in the list, a connecting edge, and its corresponding negative edge (directed graph) is created between the other nodes in the list. Then the next node in the list is processed, and so on until the end of the node's list is reached. The graph's edge lists must be checked for redundancy prior to each edge insertion. This will eliminate duplicate edges. This edge insertion process is $O(nm^2)$ for n nodes with adjacent node lists of length m . However, since n is typically $\gg m^2$ the graph creation process is effectively of the magnitude $O(n)$. This method is called BN_Create_Graph.

In an effort to detect errors in the input data and to eliminate problems within the recursive partitioning, it is necessary to ascertain that the system graph in question is "connected." This implies that all nodes are contained within a single graph and can be reached via a graph traversal from any vertex (node) within the graph. If the graph is disjointed before partitioning there is an error in the input data and the algorithm is aborted.

As stated above, the partitioning algorithm is recursive. Therefore, when a graph is partitioned, the partitions or sub-graphs are resubmitted to the algorithm for further partitioning. Each sub-graph obtained is tested for connectivity before partitioning is repeated. Unconnected partitions are undesirable and can lead to numerical instabilities within the problem.

The methodology utilized to determine graph connectedness is a form of the so-called "Disjoint-Set" ADT, implemented with union-by-rank and path compression. The worst case complexity for this configuration is very nearly linear, $O(n)$. While occurrences of non-connectedness are rare, the possibly severe consequences justify its use. Also, with

consideration of Requirement 1, the handling of irregular topologies, graphs which have odd connectivity patterns need to be checked for loose, or disconnected vertices. These can be corrected before further partitioning is undertaken.

As noted above, the graph configuration enables $O(n)$ traversals of the vertices. Depth first traversals provide a method for measuring relative distances within a graph structure. The traversal provides a quantitative measure of hops between vertices. Also, it doesn't measure in a straight line, but instead "fans" out away from the source. This provides a computationally economical method for determining measurements within the graph. Traversals are not limited to measurement from a single reference vertex. They can also be used to measure from multiple references simultaneously.

Traversals from multiple reference points provide a means of identifying which node(s) are furthest from the reference nodes as a group. For example, when the traversal is initiated at each of the four corners of a symmetric two-dimensional graph, denoted a-d, the traversal proceeds simultaneously from each corner. Interestingly, the traversal intersects along the lines where quarter sectioning should take place. If the nodes associated with their respective corners, a-d, are removed as groups, they create four identical sub-graphs. What this traversal has done is to provide a means of marking the centers between the corners and, in a manner of speaking, defines cut lines that demarcate conglomerate structures of equivalent computational effort.

It appears that this procedure yields successful partitions in both two and three dimensions. The only caveat being how the corners are found. What is needed then is a methodology that finds the corners, or "extreme points" of the graph. These extreme points, or corners, are called "seeds" throughout this discussion. This is because they provide the seed points for the multi-traversals, also known as "seed-traversals." Once the proper "seeds" are established, it is a trivial matter of a multi-point seed-traversal to identify the partitions associated with the seed points.

For some insight into this, recall the original goal is bisecting the maximum graph diameter. Assume that the largest possible original v.reach for a given graph is known. This quantity represents the furthest possible direct distance between any two nodes within the graph and is denoted max_reach. The goal is to find the so-called seed(s) such that the maximum v.reach after a multi point traversal with the seeds is equal to half that of max_reach. The longest v.reach (max_reach, obtained via a single-point traversal) associated with a given graph must be halved, as indicated by the maximum v.reach obtained via a multi-point seed traversal. Satisfaction of this criterion guarantees that no partition will have a max_reach greater than one-half the original. This is equivalent to halving the maximum diameter of the graph since the diameter is simply a measure of the mean v.reach of a partition. Thus,

$$\max_reach_{partition} \leq \frac{1}{2} \max_reach_{original} \Rightarrow \beta_{partition} \leq \frac{1}{2} \beta_{original} \quad \text{Eq. 28}$$

where β represents the diameter of the system.

Simply put, the process described consists of two distinct tasks:

1. Identification of $max_reach_{original}$ and
2. Finding sufficient seed points to satisfy the relationship for (28).

Identifying max_reach is a matter of finding the longest distance between any two nodes (vertices) in the graph. The longest distance will obviously be between points on the periphery of the graph. Therefore, the first step is to ensure that measurements (traversals) are taken from vertices (nodes) on the graph perimeter.

Given a random starting vertex, conduct a single point traversal from that vertex. The node(s) with the largest $v.reach$ associated with that traversal must lie on the periphery. Arbitrarily choose one of those vertices, $v_{periphery}$, and conduct another single-point traversal from it. The largest $v.reach$ associated with this traversal is defined as $max_reach_{original}$. It is the diameter of the graph.

The seed points for all geometries need to be identified. To this point, we have been dealing with example uniform graph topologies with clearly demarcated boundaries. It is intuitively obvious that the four corners of a square and the eight corners of a cube represent the extreme points of their respective geometries. What about irregular topologies? This methodology needs to identify the not so obvious seeds (extreme points) in all cases, not just the ideal ones. Since the seed points represent the extreme vertices in the graph, the following observations can be made:

- Seeds always lie on the graph periphery; and
- Seeds represent vertices that are furthest from most other vertices.

The first observation is intuitive. If the definition of a seed point is that of an extreme, or corner point, it must lie on the outer boundary. Any vertex on the interior of the graph always has a neighbor, which is further away, towards a boundary. Thus, since a seed is the furthest neighbor, it can't lie on the interior of the graph.

This section has provided some background information on graph structures and developed a strategy for performing the graph partitioning. The following section details a straight-forward algorithm to illustrate the concepts delineated.

Partitioning Algorithm Analysis

Figure 41 provides a formal description of the algorithm. The complexity of each line is given. Line 1 is a test to see if further partitioning is warranted. The cutoff test may be as simple as checking the number of vertices in the graph. For instance, less than X number of FLOPs in a partition may be undesirable due to the time for context switching in the machine. The test for disjoint sets is Line 2. Its complexity is very nearly linear as discussed above.

Finding the longest dimension requires two traversals of the system graph, as reflected on Line 5. Line 7 represents the number of seeds required to bisect the graph diameter max_reach . This is also the number of resultant sub-graphs, or partitions. As discussed earlier, the maximum number of sub-partitions occurs in a three-dimensional topology, which results in a maximum of eight (8) sub-graphs, therefore eight (8) seeds are the

most required per partition. After the initial traversals to find the longest dimension, as in Line 5, subsequent traversals are initiated only from vertices on the periphery of the graph, as indicated by Line 8. The number of vertices on the periphery varies by graph topology.

The candidate seed nodes are limited to those extreme points in the graph, which exist at the ends of the longest dimension within the graph. Hence, with problems of high aspect ratio, such as a one-dimensional string of vertices, candidate seed nodes exist only at the opposite ends of the graph, although there are periphery nodes all along the string. Graphs with aspect ratios approaching unity have the highest incidence of seed candidates. For instance, consider a two-dimensional, square shaped graph, with a total of n -nodes. The possibility exists that a traversal must be conducted for every node on the periphery to find each seed. For the square case, this represents $O(4n^{1/2} - 4)$ vertices. However, a denser, three-dimensional configuration has a larger ratio of surface to interior nodes. The cubic case has $O(6n^{2/3} - 12n^{1/3} + 8)$ vertices on its outside surface. This relationship, while probably overly conservative, is used, since the actual number required depends on unpredictable, random factors. In actuality, the rate of finding seeds increases with the number already located.

Lines 9 – 12 are functions performed up to Line 8 number of times, until a single candidate is found, which is then identified as a seed node. Together, these lines represent a complexity of $O(3n + 1)$. Likewise, once a seed has been determined, Lines 13 – 17 are performed once for each seed in the graph, (Line 7 number of times) representing a complexity of $O(3n + 2)$ operations.

Once all of the seeds have been located, the graph is divided into as many partitions as seed nodes. The complexity of this operation is $O(n)$. Then, the sub-graphs are recursively submitted to the algorithm for further partitioning, as denoted in Line 21. This process is described as $cT(\frac{n}{c})$, where c is the number of seed nodes. The complexity of the entire algorithm can be simplified into the following form:

$$T(n) = cT\left(\frac{n}{c}\right) + cO[(6n^{2/3} - 12n^{1/3} + 8)(3n + 1) + (3n + 2)] \quad \text{Eq. 29}$$

Further reduction yields the following recurrence relation

$$T(n) = cT\left(\frac{n}{c}\right) + O(n^{5/3}) \quad \text{Eq. 30}$$

Utilizing the Master's Theorem (Cormen, Leiserson and Rivest 1995) for solving the recurrence relationship yields

$$T(n) = \Theta(n^{5/3}) \quad \text{Eq. 31}$$

since the polynomial term $n^{5/3}$ dominates.

```

Int BN_PARTITION( BN_Graph g )
{
    unsigned int max_seed_reach, max_reach;
    stack candidate_stack, seed_stack, child_stack;
    BN_Graph child;
    vertex v, seed;

1   if(i_meet_base_case(g)) return 1; //  $O(n)$ 
2   if(i_am_disjoint(g)) //  $O(n)$ 
    {
3       error("Graph not connected"); //  $O(1)$ 
4       return 0; //  $O(1)$ 
    }

5   max_reach = find_longest_dimension( candidate_stack, g ); //  $O(2n)$ 
6   max_seed_reach = max_reach; //  $O(1)$ 

7   while(max_seed_reach > (0.5 * max_reach)) //  $O(8)$ 
    {
8       while(candidate_stack.size != 1) //  $O(6n^{2/3} - 12n^{1/3} + 8)$ 
        {
9           v = pick_random_candidate( candidate_stack ); //  $O(1)$ 
10          single_point_traverse( g, v ); //  $O(n)$ 
11          update_current_weight( g ); //  $O(n)$ 
12          identify_new_candidates( candidate_stack, g ); //  $O(n)$ 
        }

13         seed = pop( candidate_stack ); //  $O(1)$ 
14         push( seed_stack, seed ); //  $O(1)$ 
15         multi-point_traverse( g, seed_stack ); //  $O(n)$ 
16         max_seed_reach = find_max_weight( g ); //  $O(n)$ 
17         identify_new_candidates( candidate_stack, g ); //  $O(n)$ 
    }

18   child_stack = divide_graph( g, seed_stack ); //  $O(n)$ 

19   while(child_stack.size != 0) //  $O(c)$ 
    {
20       child = pop( child_stack ); //  $O(c)$ 
21       BN_PARTITION( child ); //  $T(n/c)$ 
    }

22   return 1;

```

Figure 41: BN_PARTITION Algorithm

BN_Graph Reintegration

Reintegration can be defined as the reverse process of decomposition, or partitioning. BN_PARTITION creates a hierarchical partitioning of a system mesh that can be represented as an BN_Tree, T . The partitioning process sub-divides the original, DAG structure. The resultant BN_Tree is a map of the decomposition process.

Each substructure, S , within the tree has a pointer to its parent structure, and a deque of pointers to its children structures. A substructure utilizes this information to transmit information to and from its children, and parent structure. In this case, note that data may be passed both up and down the tree. Down when calculating BN statistics, and up when updating tables based on an instantiation of evidence. The table updating process of the problem begins at the leaves, or $S.level=1$ structures, and progresses up the tree. The statistic calculation phase begins at the root, and works its way down to the tree leaves. This is how the pieces that were divided by partitioning are recombined. This assembly process is known as “reintegration.”

The structure of the BN_Tree defines the reintegration process. Heretofore, the partitioning process has defined the structure of the BN_Tree, and the concomitant reintegration. For ideal problems, with 2-D square topologies this is rightfully so. However, irregular meshes that are characterized by high-aspect ratio may have redundant upper levels in the BN_Tree. This wastes memory and computational effort.

Elimination of the unnecessary upper levels of this high-aspect ratio problem rectifies the problem of redundant storage, and reduces overall computational effort. This situation is depicted in Figure 42. Note that the structure is fully reassembled at Level 2, instead of reassembling the structure as it was decomposed. This hasn't changed the nature of the lowest level partitions, which bisected the diameter of the original mesh. It simply eliminated the levels where the diameter wasn't decreased.

Reintegration Algorithm Development

The desired result of BN_Reintegration is to effectively eliminate partitions, which are redundant, from the BN_Tree. Each substructure, S , has $S.diameter$ and $S.comp_effort$ (computational effort for evaluation) defined. Figure 42 illustrates a schematic of the proposed reintegration process. Redundant substructures, S_2 and S_3 , are removed from the BN_Tree. $S_{4,7}$ are now the children of substructure S_1 . This action has changed the fundamental makeup of S_1 . It is now assembled from four different child structures. Therefore, the connectivity table entries and communication paths that comprise S_1 have changed.

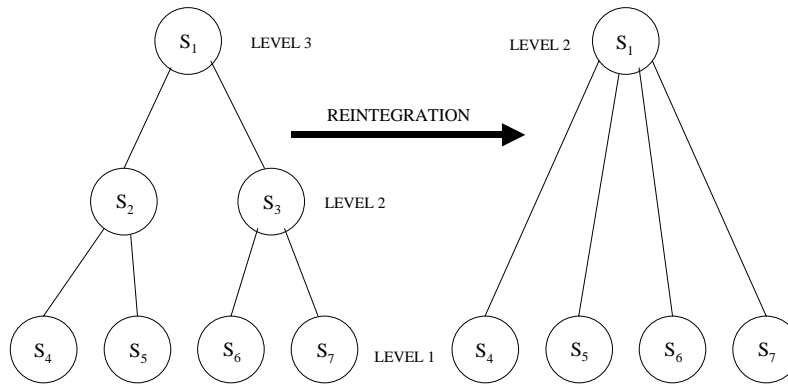


Figure 42: BN_Tree Schematic of BN_Reintegration

In effect, the BN_Reintegration performs “tree compression” on the BN_Tree. The algorithm must perform the following tasks to remove the extraneous levels.

Identify levels within the BN_Tree which are characterized by high aspect ratios.

Determine whether the structure should be removed from the BN_Tree, via some quantitative property. If so, remove the structure from the tree, and connect *S.children* to *S.parent*. Recalculate diameter and computational effort parameters for *S.parent*.

Step one requires identification of structures with high aspect ratios. Since the algorithm has no way of intuitively viewing a partition, nor of easily calculating the aspect ratio, it must assume that each substructure in the tree, *S.Level > 1* is a candidate for removal. Thus, all *S.Level > 1* structures, excepting the root, may be removed if they meet the proper criterion. This implies that a BN_Tree must have at least three levels in the original tree for reintegration to be utilized. This is so because compressing a level from a three-level tree leaves only two levels. Compressing a two-level tree implies that only one level remains, which obviates the concept of partitioning. A BN_Tree must have at least two levels otherwise the partitions can’t be reassembled.

The second step requires the establishment of quantitative criteria for determining whether a substructure is removed from the tree. Viewed objectively, the basic question to be answered is, “Will the BN_Tree be better off without the substructure, *S*, in question?” Since the goal of partitioning is to reduce the computational effort of the solution process, the parameter *S.comp_effort* will be used to make this determination. It provides a measure of the computational effort required to factor the system matrix of *S*. It is a function of *S.diameter*, and the size of the probability table in question.

The computational effort for the entire BN_Tree must be less without *S*, than with it, to justify the removal of *S* from the tree. However, it doesn’t make sense to remove *S* from the tree prior to testing. Because if it fails the test, *S* will have to be reinserted, an expensive process to be sure. Therefore, a “candidate” structure that models the removal of *S* from the tree can be created for testing, to ascertain if removal is warranted. If so, the candidate can be easily inserted into the BN_Tree. If not, then the candidate is discarded without disturbing the existing BN_Tree.

Utilization of this candidate substructure inherently covers the third and fourth tasks previously enumerated. The candidate is actually a copy of the S_{parent} substructure, with S removed from its *parent.children* deque, and $S_{children}$ inserted. Effectively, S_{parent} takes on its grandchildren structures, $S_{children}$, and eliminates S . This candidate parent must be revalued to ascertain *candidate.diameter*, and the concomitant *candidate.comp_effort*. Assume that the substructure, S_2 , in Figure 43 is being tested for removal from the tree. The parent of S_2 is S_1 . Therefore, removal of S_2 creates a change in the children of S_1 , and its system matrix. Since S_2 and S_1 will be affected by the removal of S_2 , their sum total computational effort is the standard for comparison against the candidate. S_{3-7} do not change, so $S_{3-7}.comp_effort$ need not be considered in the testing process. Hence,

$$Comp_effort_{original} = S_2.comp_effort + S_1.comp_effort \quad \text{Eq. 32}$$

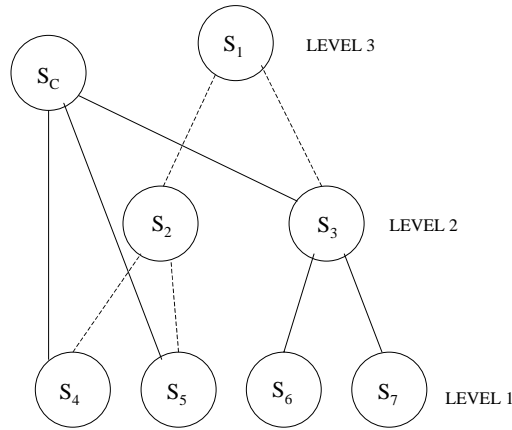


Figure 43: Test Implementation of Candidate Structure

The setup for the candidate substructure, S_C , is depicted in Figure 43. The candidate structure takes the place of S_1 , since it is the affected structure. The dashed lines indicate that the pointers to S_2 and S_1 have been temporarily redirected to the candidate structure. Once the candidate test structure, S_C , has been implemented, it is necessary to determine $S_C.diameter$, and $S_C.comp_effort$. If

$$S_C.comp_effort < Comp_effort_{original} \quad \text{Eq. 33}$$

Then, the candidate structure is implemented as depicted in Figure 44, $S_1' = S_C$, S_2 and the original S_1 are thrown away. Otherwise, if

$$S_C.comp_effort < Comp_effort_{original} \quad \text{Eq. 34}$$

The temporary pointers to the candidate structure, S_C , are redirected to their original locations, and S_C is deleted.

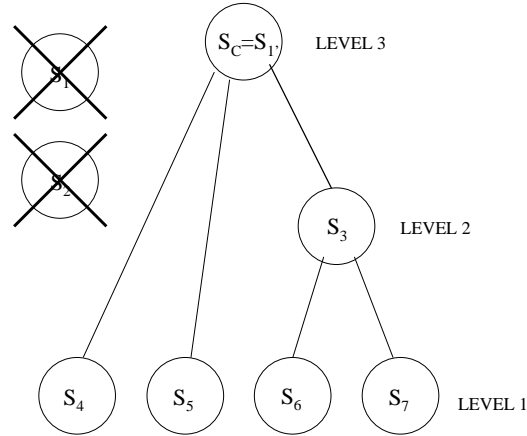


Figure 44: Permanent Insertion of the Candidate Structure

Assume the initial tree has been traversed, and each substructure has been tested for removal. If the tree has not changed from its original state, no reintegration was needed within the tree and the algorithm stops. However, if one or more substructures within the BN_Tree were removed, the entire tree must be resubmitted for reintegration testing. This process continues until the size of the BN_Tree, $T.size$, hasn't changed for two cycles. This implies that all the unnecessary partitions have been removed from the tree. The rationale for this iterative submittal is intuitive. Assume that the structures within T are arranged for testing according to a level-order traversal of T . This means that the structures are tested for removal from the top-down. If the tree depicted in Figure 44 is a sub-tree of another, larger tree, then S_1 would have a parent structure as well. $S_{1.parent}$ was tested for removal based upon the original configuration of S_1 . However, with the insertion of the candidate, S_C , the parameters of S_1 have changed to $S_{1'}$. These changes may also affect $S_{1.parent}$. Therefore, the entire tree is resubmitted to BN_Reintegration to test $S_{1.parent}$ for removal based upon its new configuration. Repetition of this process is conducted until no further changes in BN_Tree are noted. This guarantees that no redundant partition exists.

Reintegration Algorithm Analysis

The preceding example serves to illustrate the development of the reintegration methodology. Figure 45 provides a more formal description of the algorithm. The complexity of each line is given.

Line 1 is the outermost loop in the algorithm. This loop forces continued attempts at partition removal until the size of the tree stays the same for two iterations. The algorithm is setup to proceed in a level-order traversal of the tree, hence if an entire level is to be compressed, it will be done in one iteration. Therefore, the maximum number of iterations to convergence is $T.max_level$. This would imply that the remaining tree has only two levels, which cannot be further compressed. The maximum possible number of levels in the tree is $O(\lg n)$ since there are at least two substructures created from each parent, with n -elements in the original BN_Graph generated via BN_PARTITION.

The loop conditional argument is reset to the current size of the tree in Line 2. If the tree stays the same size through the loop, then it will terminate as Line 1 is tested. A deque of the structures within the tree is created. They are arranged in top-down, level order. This allows the reintegration to proceed from the top of the tree. The number of structures in the tree is $O(n)$, where n is the number of elements in the original model.

```

1  int BN_Reintegration(BN_Tree T)
2  {
3      int original_size=0, original_effort;
4      deque substruc_deque;
5      BN_Sub S, candidate;
6
7      1 While(T.size!=original_size) //  $O(\lg n)$ 
8      {
9          2 original_size=T.size;
10         3 Substruc_deque=level_traversal(T); //  $O(n)$ 
11
12         4 while(substruc_deque.size!=0) //  $O(n)$ 
13         {
14             5 S=popdeque(substruc_deque); //  $O(1)$ 
15             6 If(S.level>1 && S.level<T.max_levels) // $O(1)$ 
16             {
17                 7 Original_effort=S.comp_effort+S.parent.comp_effort; $O(1)$ 
18                 8 Candidate=create_candidate(S); // $O(1)$ 
19                 9 BN_Update(S); // $O(nm)$ 
20                 10 If(candidate.comp < original_effort) // $O(1)$ 
21                 {
22                     11 integrate_the_candidate(candidate);//  $O(1)$ 
23                     12 dispose(S);//  $O(1)$ 
24                     13 dispose(S.parent);//  $O(1)$ 
25                 }
26             }
27             14 else
28             {
29                 15 dispose(candidate);//  $O(1)$ 
30             }
31         }
32     }
33
34     16 return 1;
35 }

```

Figure 45: BN_Reintegration Algorithm

All substructures in the deque are emptied out in the while loop of Line 4. Line 5 and 6 are simply constant order operations. A substructure is removed for testing in line 5. If the substructure is on the first level, or is the root structure of the tree, it is not tested for removal.

Line 7 is a constant order addition that totals the computational effort of the substructure and its parent. The candidate structure, as described above, is created in Line 8. This is a constant order operation, since it is merely a copy of the parent, with some minor alterations.

The bulk of the algorithmic computational effort is in Line 9, where the candidate diameter and computational effort is evaluated via BN_Update, $O(n^{2/3}m)$. If the effort of the candidate is less than the original (Line 10), it is integrated into the tree (Lines 11-13). Otherwise, it is disposed of in Line 15. Lines 10 – 15 are constant order complexity, and therefore do not play a role in determining the overall complexity of the algorithm.

$$O((\lg n)n(n)m) \quad \text{Eq. 35}$$

This can be simplified to

$$O(n^2 \lg n(m)) \quad \text{Eq. 36}$$

The quantity m represents the size of the manifold of the probability set within the structure being numbered. Thus, the size of m depends upon the individual problem, and the degree of reintegration. Obviously, m will grow as more levels are compressed from the tree. This means that more child structures are being combined to form candidate structures. A reasonable worst case scenario is $m \rightarrow n^{2/3}$, instead of n . This is because the lowest level structures have a finite size. An average case may be on the order of $m \rightarrow n^{1/3}$.

Algorithm Application

A satisfactory graph decomposition tree created by the algorithm outlined above is denoted as BN_Tree (multi-level substructures). It is necessary to map the sub-trees to a given distributed computing platform to effect a solution. This can be accomplished by parsing the BN_Tree into so-called “branches” for disbursement among individual processors, as shown in Figure 35.

The edges between unlike branches of the tree represent communication between different machines in the distributed environment. The “critical path” of the solution, which is equal to the longest path from branch to root in the above tree, is a function of the computational effort (FLOPs) of each branch, message size, and network latency. (Note that each branch is processed entirely on one machine.) It represents the minimum possible elapsed time for complete solution of the given mapping in a distributed environment.

Algebraic process specifications (Baeten and Weiland 1990) offer a convenient notation to represent the cost function as implied by the BN_Tree. As an example, consider the BN_Graph shown in Figure 46. In this case, the arrows illustrate the data paths, and note that data may be passed both up and down the tree. Down when calculating BN statistics, and up when updating tables based on an instantiation of evidence.

The updating sequence for the BN_Graph requires “forward” solution processing of each grain going up the tree, and the BN chain rule of calculation invokes processing down the tree in a level order fashion. As mentioned above, concurrency is allowed between individual grains on each level. We allow the process of grain E on the forward solution to be represented as E, and the process of grain E on the back-substitution phase to be denoted E'. Utilizing similar notation for each grain, the sample elimination tree of Figure 46 can be expressed with process algebraic notation techniques as the following expression:

$$(E \parallel F \parallel G \parallel H \parallel I \parallel J \parallel K); (B \parallel C \parallel D); (A; A'); (B' \parallel C' \parallel D'); (E' \parallel F' \parallel G' \parallel H' \parallel I' \parallel J' \parallel K') \quad \text{Eq. 37}$$

Subscripts can be used to denote process dependencies. For example, since B must follow E and F, the expression $B_{E,F}$ conveys that process dependency. Utilizing this terminology (37) can be recast as:

$$(E_{\text{null}} \parallel F_{\text{null}} \parallel G_{\text{null}} \parallel H_{\text{null}} \parallel I_{\text{null}} \parallel J_{\text{null}} \parallel K_{\text{null}}); (B_{E,F} \parallel C_{G,H} \parallel D_{I,J,K}); (A_{B,C,D}; A'_A); (B'_{A'} \parallel C'_{A'} \parallel D'_{A'}); (E'_{B'} \parallel F'_{B'} \parallel G'_{C'} \parallel H'_{C'} \parallel I'_{D'} \parallel J'_{D'} \parallel K'_{D'}) \quad \text{Eq. 38}$$

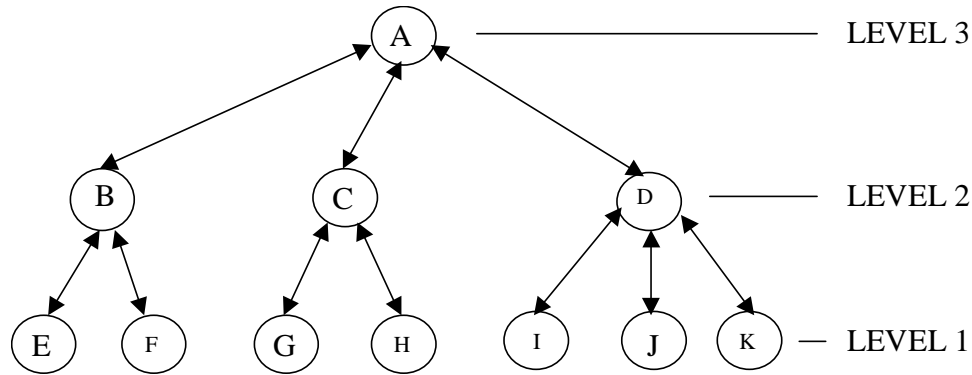


Figure 46: Sample BN_Graph Elimination Tree

Using the so-called “shuffle” function of Stach 1995), and the axioms of process algebra, (37) can be recast in a variety of ways. For instance:

$$((E_{\text{null}} \parallel F_{\text{null}}); B_{E,F}) \parallel ((G_{\text{null}} \parallel H_{\text{null}}); C_{G,H}) \parallel ((I_{\text{null}} \parallel J_{\text{null}} \parallel K_{\text{null}}); D_{I,J,K}); (A_{B,C,D}; A'_A); (B'_{A'} \parallel C'_{A'} \parallel D'_{A'}); (E'_{B'} \parallel F'_{B'} \parallel G'_{C'} \parallel H'_{C'} \parallel I'_{D'} \parallel J'_{D'} \parallel K'_{D'}) \quad \text{Eq. 39}$$

Expansion and contraction of the parenthetical groupings to combine individual grains can be used to map a BN_Graph to a particular distributed network configuration. For example, at one extreme, with only one processor available, the BN_Graph is represented by the following expression:

$$((E_{null}, F_{null}, G_{null}, H_{null}, I_{null}, J_{null}, K_{null}); (B_{E,F}, C_{G,H}, D_{I,J,K}); (A_{B,C,D}; A'_{A}); (B'_{A'}, C'_{A'}, D'_{A'}); (E'_{B'}, F'_{B'}, G'_{C'}, H'_{C'}, I'_{D'}, J'_{D'}, K'_{D'})) \quad \text{Eq. 40}$$

The “;” represents a “don’t care” condition, in which the process order is irrelevant as long as they are all completed before the “;”, even though they must be computed serially due to physical constraints. This saves permuting the sequence using the OR boolean symbol “+” for example, $a, b = (a; b + b; a)$. Assuming one machine is available for each parenthesized grain and that each machine has one processor, expression Eq. 39 can be recast as

$$((E_{null}, F_{null}); B_{E,F}) \parallel ((G_{null}, H_{null}); C_{G,H}) \parallel ((I_{null}, J_{null}, K_{null}); D_{I,J,K}); (A_{B,C,D}; A'_{A}); (B'_{A'}; (E'_{B'}, F'_{B'})) \parallel (C'_{A'}; (G'_{C'}, H'_{C'})) \parallel (D'_{A'}; (I'_{D'}, J'_{D'}, K'_{D'})) \quad \text{Eq. 41}$$

In the case each machine has sufficient multiple processors, Eq. 39 can be left intact. Use of these methods helps clarify and establish realistic cost functions, which facilitate heterogeneous network mappings for solution of BN_Graphs in a practical computing environment.

The size of individual messages depends on the number of common entries in the statistical tables, which exists between the two substructures represented by the tree nodes in question. Network latency must also be considered, and is tightly coupled to the message size. However, it is difficult to model with any accuracy since, in any circumstance other than benchmarking, the user faces contention and has no control over network characteristics.

A large-scale sparse BN graph model with > 900,000 vertices was generated. Each vertex was interconnected with a random number of neighboring vertices with maximum direct connectivity of 27 neighbors (shared table entries). Figure 47 shows a surface plot of critical path lengths vs. processor/branch totals, for a simple network model (latency directly proportional to message size) for the model.

The values on the surface indicate the “critical path” of the problem which, as previously stated, is a direct indication of the problem’s total solution time in a distributed environment. The surface value includes both computational effort (FLOPs) for updating of the statistical inference matrix, and communication latencies. The number of processors ranges from 1 to x (maximum number of Level 1 substructures). The network latency increases as denoted by the arrow, while the computational effort stays constant. The latency is modeled as directly proportional to the message size. The network model also assumes a homogeneous environment.

The plot shows that the number of processors/branches utilized drastically affect the time required for problem solution. This interesting situation arises from the communications overhead, which differs dramatically, depending upon the physical groupings of boundary substructures between branches. Using between 30 and 117 processors actually increases the solution time, compared to fewer processors, no matter what the network latency. This is due to inefficient branch structures created by forcing the problem to fit

x number of processors. This plot serves to underscore the fact that the topology of the original, top-level, system graph and its decomposition must define the network solution configuration. This illustrates that definite minimums exist.

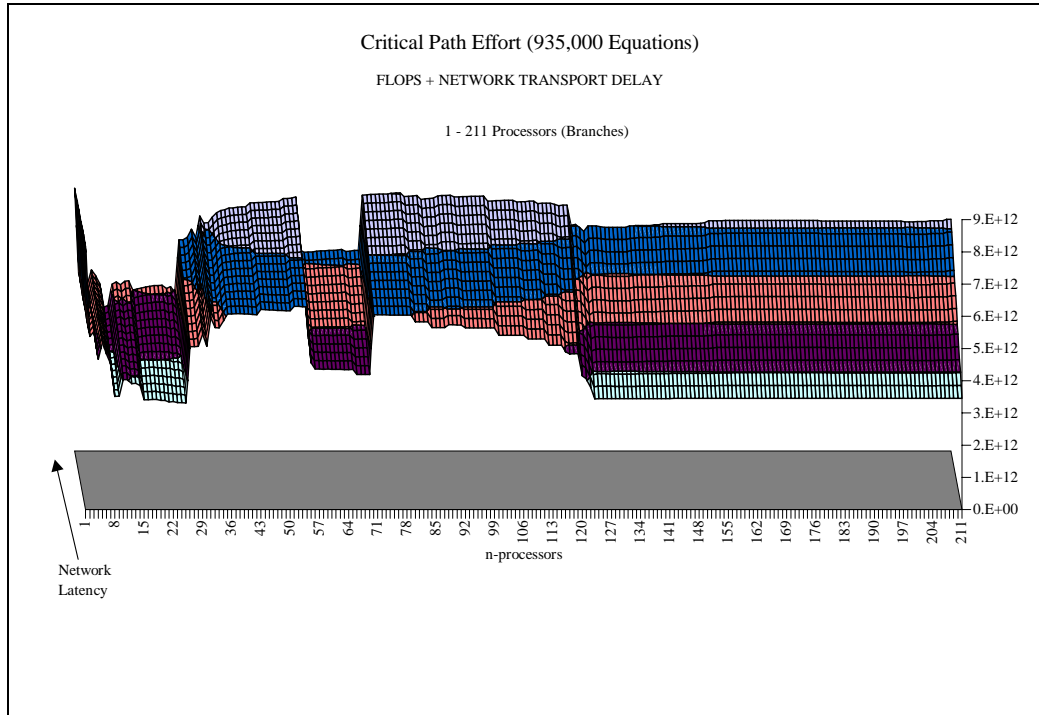


Figure 47: Example BN Graph Critical Path Effort

Figure 48 is a subset of the above plot, and shows that the most efficient number of processors, for the randomly generated problem, is nine. With nine processors the critical solution path exhibits stability throughout a wide range of network latency. This indicates that communications overhead is low with a nine branch configuration. Using 13 – 25, or 130+ processors will provide almost the same solution time.

These graphics illustrate the topology-dependent optimalities. The cost function utilized to generate them assumes homogenous machines in the distributed network. In addition, it is assumed that the network links between machines are homogenous as well.

Expansion of this research can help devise more accurate models for determining optimal network configurations. This would necessarily include accommodation for heterogeneous nodes and links in the network within the cost function. For example, the distributed solution platform in question may include supercomputers and/or workstations of varying capacity. The supercomputer may have a Rapid Array interconnect (~4 Gb/sec) connection to the mass storage unit, while the workstations may

be limited to fast Ethernet (802.3 at 1000 Gb/sec). In such a scenario, the tree parsing would obviously be weighted towards the supercomputer.

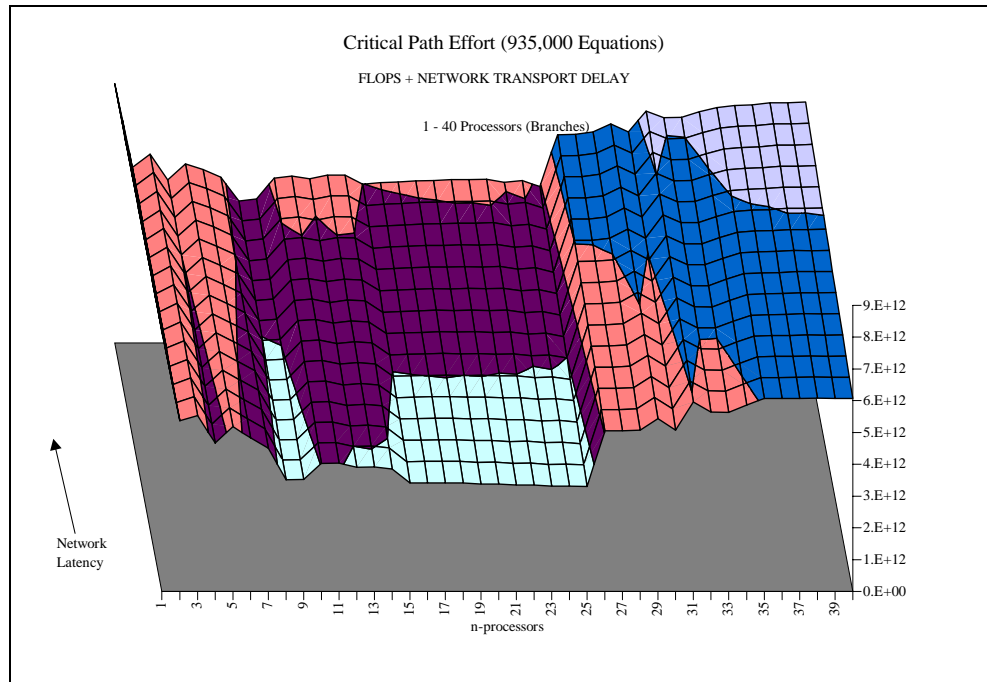


Figure 48: Sub-Plot of BN_Graph Problem

Conclusions of BN-Graph Algorithms

This section discussed the development of a Bayesian Network graph partitioning methodology based upon bisection of the maximum graph diameter. The constraining boundaries were discussed and examined for their roles' in shaping the partitioning scheme. Background information on the graph data structure and traversal methods, and their applicability to BN-graphs was presented.

An intuitive method of partitioning was presented. The resultant algorithm, BN_PARTITION, was presented more formally in a pseudo-code style format. Its recursive implementation was analyzed for algorithmic complexity. The resultant complexity is $O(n^{5/3})$. The resultant partitions from the algorithm are of uniform size, (considering the geometric irregularities), and share minimal interfacial zones, which should help reduce communications overhead over arbitrary decompositions. Thus, the four goals for the partitioning algorithm are satisfied.

Also discussed were the effects of tree compression, or reintegration, techniques on BN_Trees, which are characterized by high-aspect ratios. An approach to identify partitions that should be marked for removal from the trees was developed. The resulting algorithm, BN_Reintegration, was detailed. The main premise is to create a "candidate"

substructure that models the resultant tree if the substructure in question were removed. If it is found that the candidate improves the characteristics of the tree, i.e. reduces computational effort, then the offending substructure, and its immediate parent, is removed from the tree. Then, the candidate tree is inserted in their place. This process is conducted on all structures with *S.Level*>*I*, excepting the root structure of the BN_Tree. The tree is submitted for compression in an iterative fashion to guarantee that all unnecessary partitions are removed.

The research conducted for this study clearly indicates that Bayesian statistics can be used in military applications, particularly with respect to failure analyses. The mathematics for Bayesian statistics is very mature and proven. It is used in research as diverse as hostile missile identification to filtering spam email. The algorithms can be developed and built in such a generic fashion that the variables can be input to describe macroscopic to microscopic phenomena. As events occur, current streaming data can be input to the system making the algorithms more accurate with each updating scenario.

Furthermore, this research is cognizant of the trends toward distributed computing and the data management obstacles that must be overcome to achieve efficient, coherent solutions to large-scale BN problems. Solutions to problems of this magnitude require a high degree of data organization, efficient use of computing resources, and the flexibility to accommodate new technology. This work forms the foundation for a highly structured, object-oriented, data management methodology, which offers portability, scalability of function, and access at multiple levels of abstraction.

Electro-Mechanical Diagnostics/Prognostics

Introduction

Improvement in system reliability and reduction of wear on components is a large field of study in engineering that is normally approached from a macroscopic and/or statistical perspective. A number of processes work in concert to increase the disorder of the system, including; quantum effects, diffusion, dislocation segregation... These irreversible processes initially occur at a microscopic, or even at atomic level, (Whaley, 1984) and (Sobczyk, 2004), and are thus difficult to detect. Typically, the effects (from an engineering perspective) of these low-level processes are not evident until they reach significant proportion. One of the most common causes of failure in both electrical and mechanical components is due to stress-cracking in the system materials. These cracks are the inevitable result of low level processes acting in concert on the system materials. With time, and under fatigue stress, these cracks grow, and are manifested as a variety of undesirable effects. They can cause a variety of symptoms, including; intermittent connections on circuit boards, failure of solder joints, cracking of mechanical parts... If left unchecked, they can lead to dramatic fracture failures (Chung, 1988).

Even under static loading situations, material deterioration due to diffusion, corrosion, and creep eventually allows cracking to initiate. But, for components that exist in dynamic and rotating machinery, such as rotorcraft and other military systems, the effects are dominant. Continuous vibrations excite system natural frequencies, and dramatically accelerate the failure process (Askeland, 1989).

Real-time analysis and tracking of characteristic natural frequencies can provide an indication of crack initiation and growth in system components – conceivably prognosticating imminent FAULT conditions (Wu, et.al., 2004). Low-power, wireless, “smart” sensors demonstrating this capability could be used to enable embedded real-time condition monitoring within legacy systems without invasive action to existing power and communication sub-systems.

In support of these goals, this project examines techniques for non-invasive assessment of the physical condition of system components. Techniques for sensing component fatigue and vibration modes are discussed. Effects on the component modal characteristics due to damage/deterioration from environmental and external forces are examined. A process, with low computational complexity, for detection of anomalous modal behavior due to degraded physical condition is presented.

It is a stated objective of the funding agency to investigate and adopt practices of maintaining equipment and weapon platforms based on the current physical state or condition of the systems, as opposed to following traditional regular maintenance procedures and schedules. This change, which shifts the emphasis from preventative and reactive methodologies to a more proactive based philosophy, is commonly known as Condition Based Maintenance (CBM).

With this philosophy in mind, this project examines techniques for non-invasive assessment of the physical condition of system components. Techniques for sensing component fatigue and vibration modes are discussed. Effects on the component modal characteristics due to damage/deterioration from environmental and external forces are examined. A process for detection of anomalous modal behavior due to degraded physical condition is presented. Additionally, a prototype system for implementation of these techniques is proposed.

A generic physical component is modeled, using finite element analysis (FEA), to illustrate the effects of physical parameters and geometric boundary conditions on the general characteristic frequencies and concomitant mode shapes of the component. Two “real-world” FEA models; 1) a PCI circuit card, and 2) an example Rotor-Hub, are developed here to illustrate the capability of modal analysis to detect physical degradation of system components that commonly exist in current military systems. In addition, candidate devices and locations for sensing the component vibration modes are discussed, including strain gages and accelerometers.

The characteristic frequencies due to the vibration modes are coupled with the introduction of background noise from candidate sensing devices to create realistic models of sensor outputs. Algorithms utilizing signal processing algorithms, including the Fast Fourier Transform (FFT), and vector correlation are developed to demonstrate a method for analyzing the sensor output. This should help diagnose and prognosticate conditions potentially leading to decreased component performance.

A prototype system to implement and test these methodologies is proposed. The proposed system is based on the integration of small accelerometer sensors, field programmable gate arrays (FPGAs) and wireless radio frequency identification (RFID) technologies.

Background

Within a hybrid electro-mechanical system there are a number of mechanisms through which failures can be manifested. Furthermore, an instantiated failure presents a wide spectrum of real-world consequences, ranging from benign to potentially lethal. A FMECA (Failure Modes and Effects Criticality Analysis) is commonly used to specify and document the various types of system failures and couple them with a corresponding level of effect on the system performance. This can be used to classify the severity and impact of low-level system failures, (e.g. circuit card, or individual mechanical part) on the overall system health. Therefore, in order to effectively implement CBM and assess high-level system health issues, such as remaining useful life and time to next failure, accurate information on the physical state of low level components is critical.

An assumption of long term system health is that all components, both electrical and mechanical, are well designed and are capable of functioning to specification for a finite period in time. Ideally, the period begins at the initial system delivery, and continues for

some measurable (hopefully significant!) amount of time. This allows definition, or characterization of “healthy” system state, and provides a standard, to which all other system states can be compared. System states that lie outside the defined standard or boundaries of the “healthy” state are denoted as sub-optimal and are indicative of declining system health. Ideally, measurement of the level of deviation from the “healthy” state can enable CBM by; 1) providing an indication of the severity of the existing condition (prognosis of remaining life), and 2) helping to pinpoint the problem source to an acceptable level (e.g. SRU, LRU...) of ambiguity (e.g. component level diagnostics/prognostics).

The concept of state has both temporal and spatial (physical) characteristics, which implies the current state, $S_c = f(t, X)$, where t is time, and $X = [x_1, x_2, x_3... x_n]$, a set of physical parameters (e.g. geometry, material properties, electrical characteristics...). Transition from one state to another requires not only a change in t , but at least one change in X . This is illustrated by considering t and X as discrete variables, where Δt is the time between system measurements of physical parameters X , and ΔX indicates a significant, measurable change in system parameters. Consider the following two possibilities for small Δt :

Table 5: State as a function of t and X

Case	Current State	$\Delta t \cdot \Delta X$	Next State	Result
1	S_c	$1 \cdot 0 = 0$	S_c	<i>Next timestep AND no faults</i>
2	S_c	$1 \cdot 1 = 1$	S_{next}	<i>Next timestep AND fault occurrence</i>

Case 1 implies no faults occur from one time step to the next. This is the case for normal non-faulty operation, indicating that for the next set of measurements there is no fault detected in the physical parameters. Thus there is no measurable change in the current state, and the system remains in the current state. The second, Case 2, implies new measurements have occurred, and a change in the physical parameters of the system has been detected. The measurement in conjunction with the fault takes the system to a new state. Note that since the measurements only occur at increments of t , the physical parameters and detection of their change also depend on t , implying that $X = f(t)$.

Assuming a “healthy” state, S_H , exists at the time of system delivery or repair ($t=0$), then $S_H = f(t_0, X(t_0))$, and the system stays in state, S_H , for all t until a change occurs in X , at which point the system transitions to the faulty state, $S_{faulty} = f(t_{fault}, X(t_{fault}))$. Existence of a “healthy” state implies that specific, observable conditions (physical and/or electrical) exist for a finite period of time. Transition to a faulty state implies that the physical parameters of the system have degraded **AND** that the system measurements detect the degradation. Another, less obvious, conclusion that can be drawn from this discussion is that some time must elapse before a fault can occur. This reinforces the idea that faults are always time dependent. This says that even given a low probability of a specific fault, with enough time, it will eventually occur. Therefore, a robust CBM

methodology requires a mechanism for adequately observing system behavior and that these observations occur at small enough time intervals to detect degrading behavior in process.

Further evidence of the time dependent fault is provided by the 2nd law of thermodynamics, which refers to the increase in the entropy of a closed system as a function of time – systems tend toward disorder. An engineered component such as a circuit card or a machined driveline part is a great example of a closed, isolated system (closed from a manufacturing perspective). Its lowest entropy occurs when it is just finished, there is no more energy input to create order within that system, unless it is reworked or repaired. From that point forward in the component's operational lifecycle, it tends towards disorder. When the level of disorder exceeds some threshold value, the component can be classified as FAILED. Therefore, the ability to detect and track the amount of disorder or entropy in a system enables detection and possibly prediction of imminent system failure. With reference to Figure 49, the entropy of an example component is illustrated. For a period of time, the component performs as designed in the green, "Normal Operation" mode. As time passes, and the entropy increases, the component eventually enters the yellow zone of "Imminent Failure." When the component reaches this zone, the CBM system should recognize its state, and schedule the component for repair or replacement. Absent remedial action, the component will cross the threshold into the red FAILED zone.

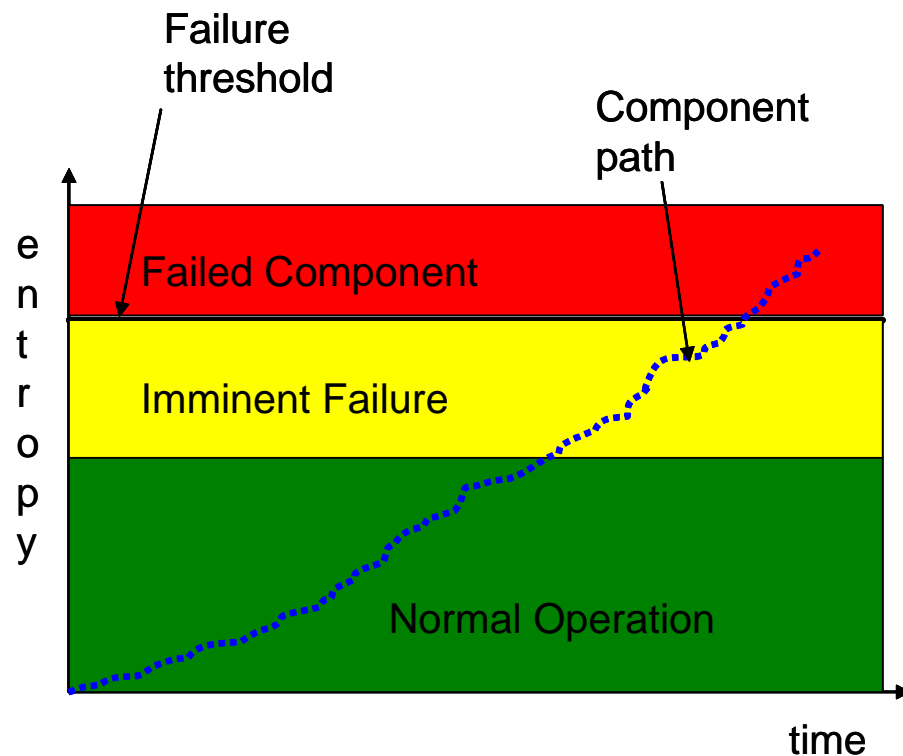


Figure 49: Example Component Path to Failure

Unfortunately, determining the entropy of systems for irreversible processes is readily quantifiable only in limited circumstances, typically only for pure substances in well defined liquid and/or gaseous phases, or in specific crystalline arrangements. Quantifying the entropy of engineering materials is much more difficult due to the lack of knowledge of internal irreversible processes (Sonntag, 1982). Hence, other indicators of disorder are sought.

Problem Definition

This research effort concentrated on modeling electro-mechanical systems utilizing the method of finite element analysis (FEA). This demonstrates the feasibility of electro-mechanical system prognostics to predict material performance degradation. The computational modeling consists of the following interrelated portions;

- Constitutive model of the composite material structure,
- Use of FEA to establish candidate locations for sensor placement,
- Sensor definition and output mode,
- Correlation of sensor data to changes in the original material model,
- Use of pattern recognition methodologies for prognoses of system health

The investigators performed analytical studies of electro-mechanical systems, operations, and diagnostic techniques and conducted research pertaining to mathematical modeling and statistical inference of failure mechanisms for electro-mechanical systems. A model of the sensor outputs was created to assess methodologies for fusing the output of the sensors into meaningful statistics to assess the probability of system failure, and concomitant remaining life estimates. This was accomplished by application of pattern recognition techniques to identify and classify system faults.

Analysis

Modal Analysis

A primary objective of the project is to develop finite element model(s) of an electro-mechanical system to determine vibration modes and frequencies. The governing differential equation for viscously damped vibration is:

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{R\} \quad \text{Eq. 42}$$

where the consistent mass matrix is defined in terms of the applicable shape functions, H and the density, ρ , of the material, which can be defined on a per element basis in the model;

$$[M] = \int_V [H]^T \rho [H] dV \quad \text{Eq. 43}$$

the so-called stiffness matrix is defined in terms of the strain displacement and material property matrices, B and D ;

$$[K] = \int_V [B]^T [D] [B] dV \quad \text{Eq. 44}$$

the damping matrix, C , can be determined from experimental modal analysis, or utilizing one of a number of analytical approximations, such as;

$$[C] = \mu [K] \quad \text{Eq. 45}$$

where the damping is proportional to the stiffness, or elasticity of the system, or

$$[C] = a[M] + b[K] \quad \text{Eq. 46}$$

which is denoted “Rayleigh” damping; and the right hand side forcing function is equivalent to the superposition of the surface, body, inertial and externally applied forces respectively.

$$\{R(t)\} = \{R_s(t)\} + \{R_b(t)\} - \{R_I\} + \{F(t)\} \quad \text{Eq. 47}$$

The solution to the governing equation is written in terms of the homogeneous and particular parts. Namely,

$$\{U(t)\} = \{U_h(t)\} + \{U_p(t)\} \quad \text{Eq. 48}$$

The natural frequencies (eigenvalues) are extracted from the homogeneous part of the solution, and are therefore functions of the material properties, geometry and boundary conditions, without regard to any particular loading condition, thus the governing equation is reduced to

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{0\} \quad \text{Eq. 49}$$

The standard solution to the 2nd order equation assumes that

$$\{u_h(t)\} = \{\phi\}e^{\lambda t} \quad \text{Eq. 50}$$

$$\{\dot{u}_h(t)\} = \{\phi\}\lambda e^{\lambda t} \quad \text{Eq. 51}$$

$$\{\ddot{u}_h(t)\} = \{\phi\}\lambda^2 e^{\lambda t} \quad \text{Eq. 52}$$

with the eigenvalues, λ , representing the natural frequencies, and the eigenvectors, ϕ , the mode shapes of the system. Substituting into (52),

$$(\lambda^2[M] + \lambda[C] + [K])\{\phi\}e^{\lambda t} = \{0\} \quad \text{Eq. 53}$$

this can be recast to facilitate the eigenvalue solution as

$$\left(\lambda \begin{bmatrix} [0] & [M] \\ [M] & [C] \end{bmatrix} - \begin{bmatrix} [M] & [0] \\ [0] & [K] \end{bmatrix} \right) \begin{Bmatrix} \lambda\{\phi\} \\ \{\phi\} \end{Bmatrix} e^{\lambda t} = \{0\} \quad \text{Eq. 54}$$

noting that in general,

$$\{\phi\}e^{\lambda t} \neq \{0\} \quad \text{Eq. 55}$$

the solution of the natural frequencies, λ , is obtained from

$$\det \left(\begin{bmatrix} [0] & [M] \\ [M] & [C] \end{bmatrix} \lambda - \begin{bmatrix} [M] & [0] \\ [0] & [K] \end{bmatrix} \right) = 0 \quad \text{Eq. 56}$$

and the eigenmodes are extracted by satisfying

$$\left(\lambda \begin{bmatrix} [0] & [M] \\ [M] & [C] \end{bmatrix} - \begin{bmatrix} [M] & [0] \\ [0] & [K] \end{bmatrix} \right) \begin{Bmatrix} \lambda\{\phi\} \\ \{\phi\} \end{Bmatrix} = \{0\} \quad \text{Eq. 57}$$

For the undamped case ($[C] = [0]$), the natural frequencies are obtained via

$$\det([K] - [M]\lambda^2) = 0 \quad \text{Eq. 58}$$

and the mode shapes by satisfying,

$$([K] - [M]\lambda^2)\{\phi\} = \{0\} \quad \text{Eq. 59}$$

for all the degrees of freedom in the system.

In practice, it is rare to solve for all the natural frequencies of large systems. Typically only the first several (≤ 10) frequencies and concomitant mode shapes are significant contributors, (Bathe, 1996) and (Thomson, 1988).

For the purposes of this project, the FE pre/post processing software Hypermesh (www.altair.com), and the finite element analysis software ABAQUS (www.abaqus.com) are utilized. The geometry and boundary conditions are built and prescribed with the HyperMesh software, while ABAQUS is utilized to construct the system matrices and solve for equations 56 – 59. Once solved, the resultant natural frequencies and mode shapes are visualized by the HyperMesh software.

As an introduction to the capability of modal analysis, the model illustrated in Figure 50 is a simple, generic component, which has available system constraints from movement at both ends A and B. The system was modeled in two different configurations, each with identical material and geometry properties; 1) by simultaneously constraining both ends A and B, and 2) by constraining only end A.

The two configurations are compared side by side to illustrate the dramatic effect of the constraints on system natural frequencies and modal shapes. Figure 51 through Figure 58 respectively depict the system mode shapes and natural frequencies of the first four vibration modalities of the two model configurations. For example, the natural frequency of the first mode shape is 55.2 Hz for the highly constrained system, and drops to 8.54 Hz for the component with reduced constraints.

Note that in each case, the model that is more tightly constrained to movement vibrates at a higher frequency than its less constrained counterpart. This phenomenon is similar to the effect seen when tightening a guitar string. A tighter string vibrates at a higher frequency (and pitch) than does the same string with less tension. These differences in vibration can be used to identify when the vibration profile of a system deviates from the expected or “normal” operational vibration profile.

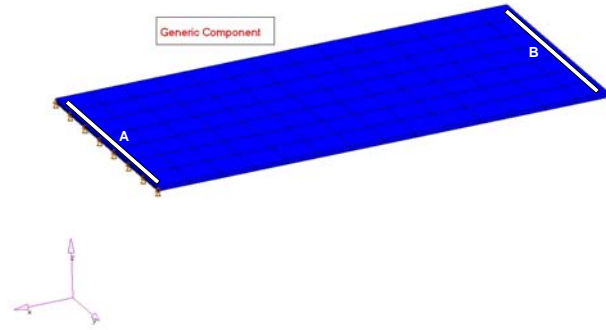


Figure 50: Generic Component, Constraints at A and B

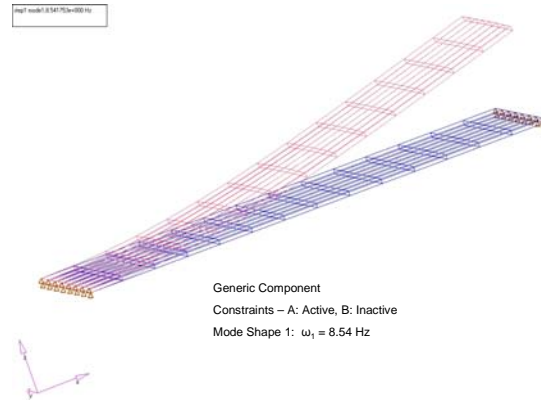


Figure 51: Mode 1, Only A Constraint Active, $\omega_1 = 8.54$ Hz

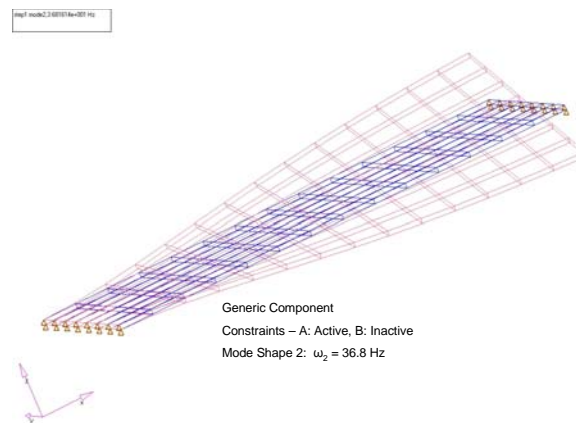


Figure 52: Mode 2, Only A Constraint Active, $\omega_2 = 36.8$ Hz

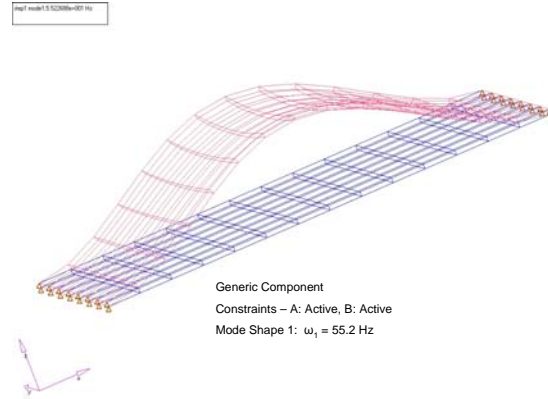


Figure 53: Mode 1, A and B Constraints Active, $\omega_1 = 55.2$ Hz

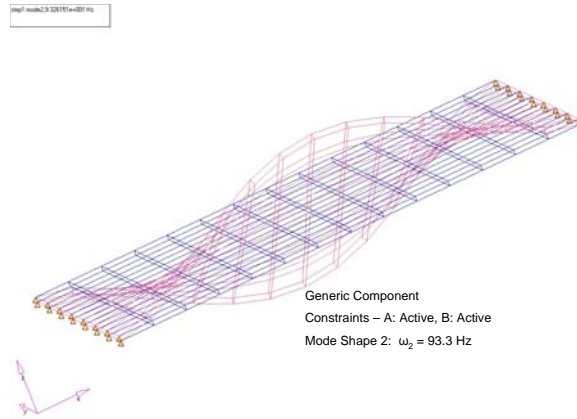


Figure 54: Mode 2, A and B Constraints Active, $\omega_2 = 93.3$ Hz

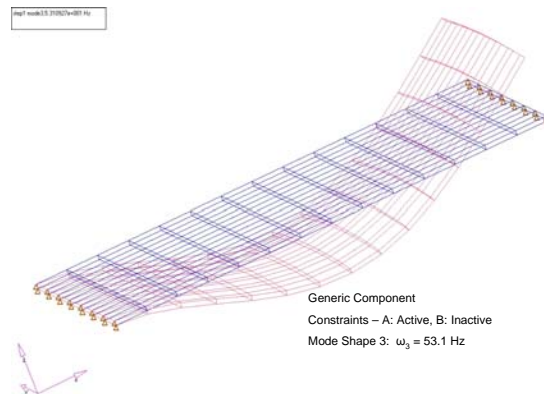


Figure 55: Mode 3, Only A Constraint Active, $\omega_3 = 53.1$ Hz

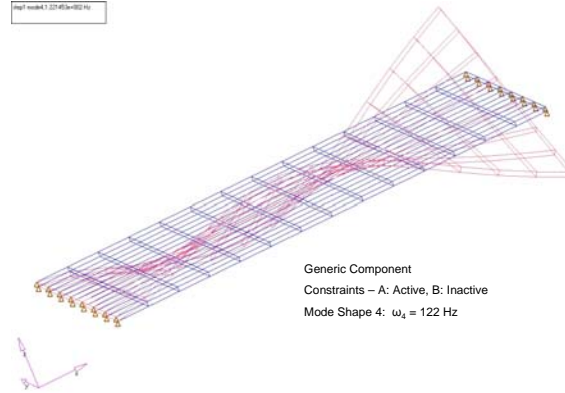


Figure 56: Mode 4, Only A Constraint Active, $\omega_4 = 122$ Hz

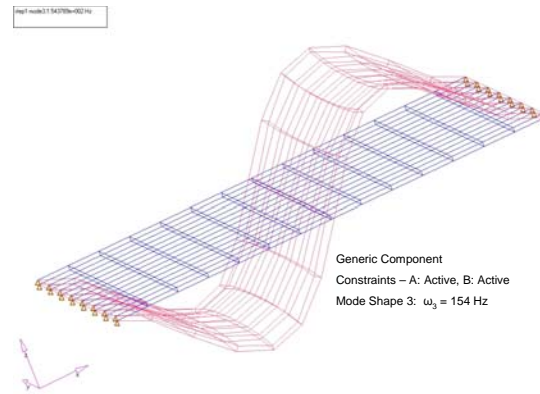


Figure 57: Mode 3, A and B Constraints Active, $\omega_3 = 154$ Hz

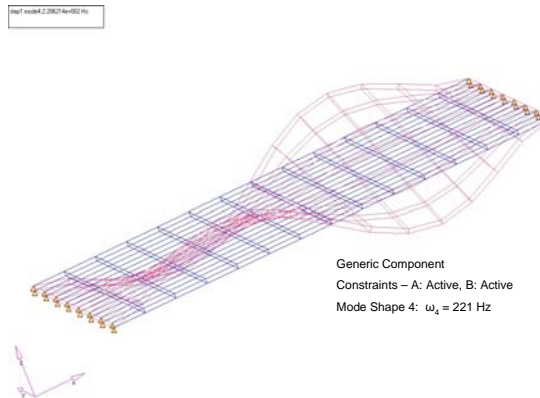


Figure 58: Mode 4, A and B Constraints Active, $\omega_4 = 221$ Hz

Sensor Selection and Placement

There are two main types of devices to consider for sensing of the vibration characteristics of system components, accelerometers and strain gages. Both sensors come in a variety of size and package configurations, some of which are small enough to be easily placed on components as small as a circuit card.

Accelerometer sensors work by providing an output voltage proportional to the detected acceleration of the sensor. They offer many benefits; low power consumption, small footprints for piezoelectric models (on the order of 2 mm²), multi-axial sensing (1 to 3 dimensions) coupled with the capability to detect a wide range of accelerations (e.g. micro-g's through 100s of g's), and many offer dynamic shock protection up to 1000s of g's.

Strain gages sensors work by sensing the change in resistance of a serpentine arrangement of wires directly attached to a component. They are hooked through a Wheatstone bridge, and provide a direct measurement of the amount of strain present in the component, which is directly related to the stress through Hooke's Law relationships. The benefit is that very accurate measurements of actual stress can be drawn from them. However, there are at least two main drawbacks that make them unsuitable for use in this situation; 1) relatively high power consumption (compared to accelerometer), and 2) the placement of the gage is critical. If the gage is improperly placed in a region of low stress, the character of the dynamic stress is lost.

In some situations, particularly on a circuit card, there may not be room to directly attach a strain gage in the high stress region, which typically occurs near the bus connection. Moving the gage to a region of the card with more available space may cause the sensor to miss localized stress tensors. Similarly, it is desirable to place accelerometers in areas on the card that undergo the most movement, but as can be seen from the modal simulations, there are a number of candidate locations. For these reasons, the accelerometer is the sensor of choice, and its placement is not as critical as that of the strain gage.

Signal Processing

Assuming an ideal accelerometer sensor, with little to no noise ratio, Figure 59 and Figure 60 illustrate appropriate time domain signals for both configurations (A constraint and AB constraints) of the generic model from Figure 50, according to the first four system natural frequencies, as

$$a_{clean} = \sin(2\pi(8.54)t) + \sin(2\pi(36.8)t) + \sin(2\pi(53.1)t) + \sin(2\pi(122)t), \quad \text{Eq. 60}$$

$$\text{and } ab_{clean} = \sin(2\pi(55.2)t) + \sin(2\pi(93.3)t) + \sin(2\pi(154)t) + \sin(2\pi(221)t) \quad \text{Eq. 61}$$

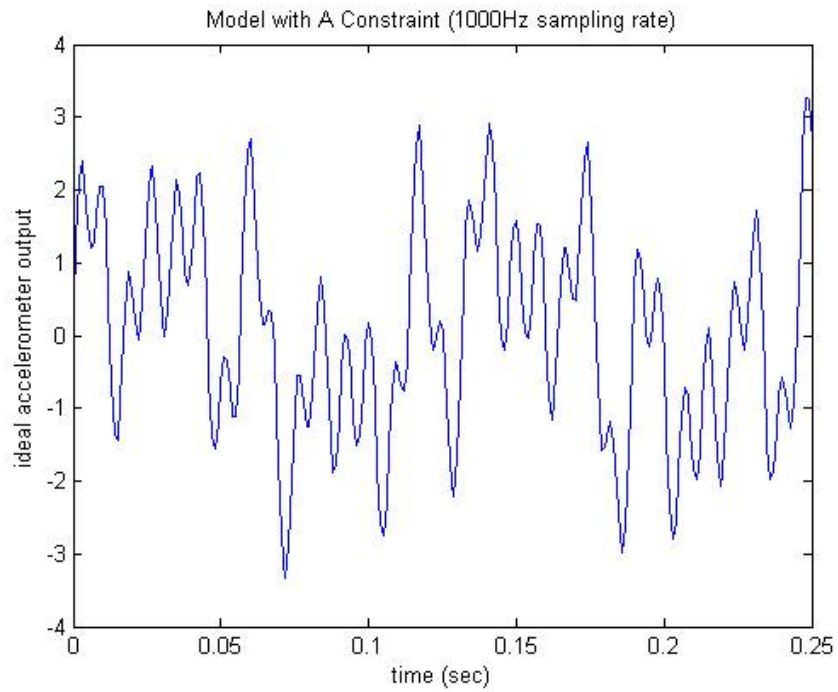


Figure 59: Generic Model, Constrained at End A, Clean Sensor Signal

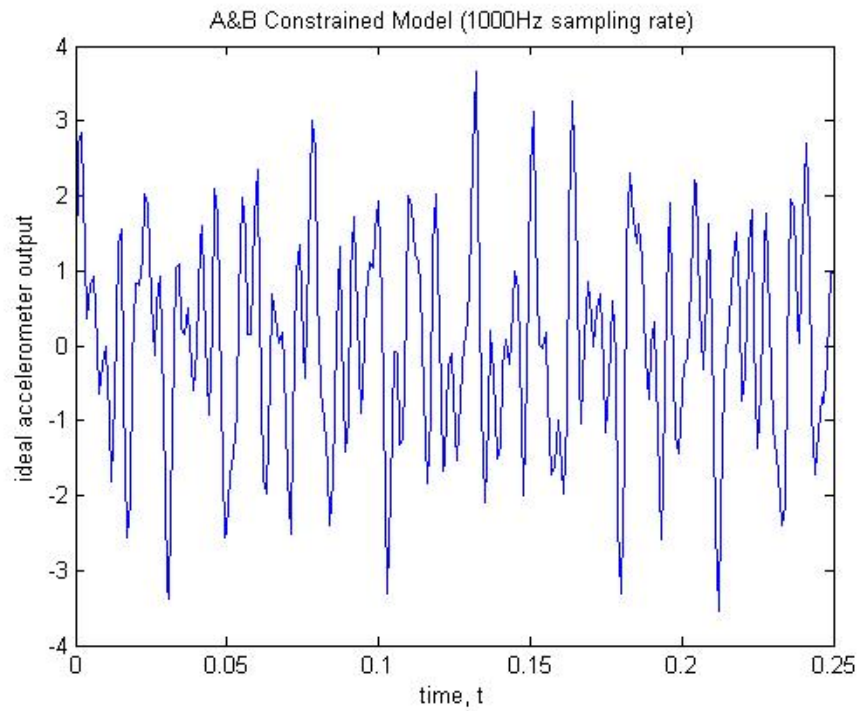


Figure 60: Generic Model, Constrained at Ends A & B, Clean Sensor Signal

The analog sensor outputs were sampled at a rate of 1 KHz for 0.25 seconds to digitize the signal as illustrated in the figures. The sampling frequency should be \geq the Nyquist frequency, which is twice the highest frequency of interest.

To simulate more realistic output conditions from the sensor model, white noise with a Gaussian distribution of amplitude, A , approximately $2/3$ of the greatest clean amplitude was introduced:

$$a_{noisy} = a_{clean} + A * random \text{ and,} \quad \text{Eq. 62}$$

$$ab_{noisy} = ab_{clean} + A * random \quad \text{Eq. 63}$$

The noisy signals created are illustrated in Figure 61 and Figure 62

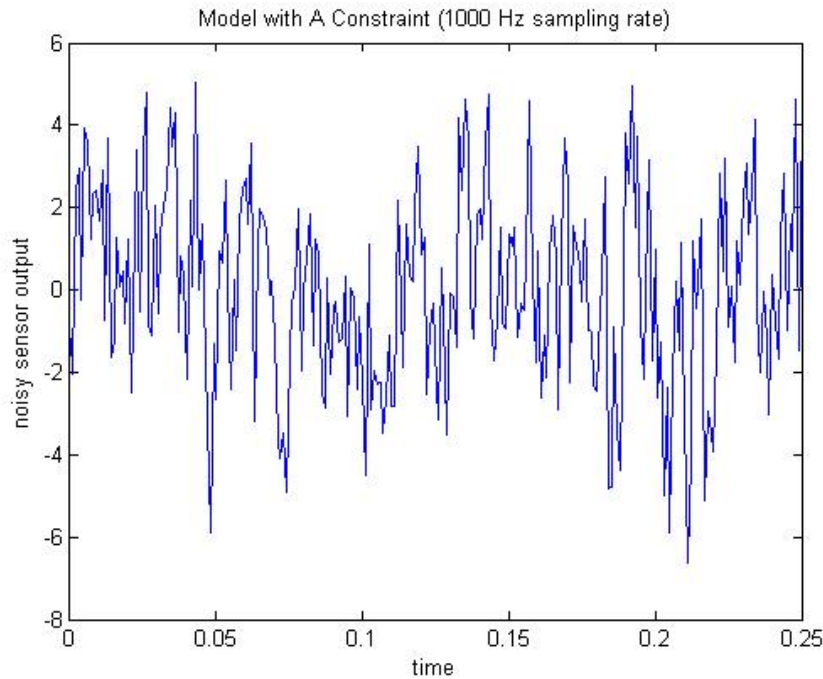


Figure 61: Generic Model, Constrained at End A, Noisy Sensor Signal

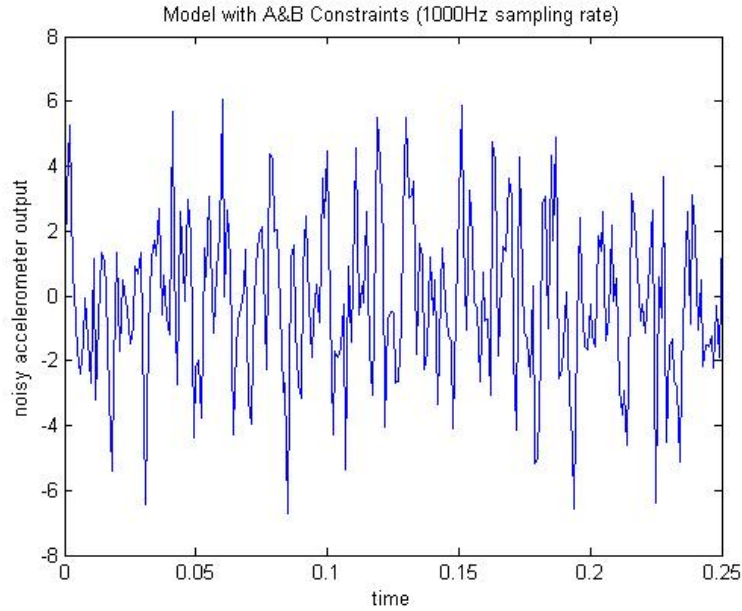


Figure 62: Generic Model, Constrained at Ends A & B, Noisy Sensor Signal

In order to compare digital signals with the goal of categorizing them as “Normal,” “Imminent Failure,” or “Failed” as shown in Figure 49, correlations to benchmark “Normal” signals can be conducted, to ascertain their degree of similarity. Those signals with a high degree of correlation are said to be similar.

An algorithm for determining the correlation, (Thomson, 1988) is given as follows:

- 1) Given two vectors of equal length, n (e.g. $x_1(t)$ and $x_2(t)$)
- 2) Multiply the ordinates of the two vectors at each timestep, i
- 3) Compute the average value of the products, for each possible combination of vector pairings:

$$\mathbf{Cor_vector}_k = \langle x_1(t) \cdot x_2(t+k) \rangle = \frac{\sum_{i=1}^n (x_{1i} \cdot x_{2i+k})}{n}; \text{ with } k = 0, \dots, n-1 \quad \text{Eq. 64}$$

- 4) Find the maximum value of $\mathbf{Cor_vector}$

$$\mathbf{Correlation} = \max \{ \mathbf{Cor_vector}_k; \text{ with } k = 0, \dots, n-1 \} \quad \text{Eq. 65}$$

An analysis of this algorithm indicates a complexity $O(N^2)$. Note that high correlation values are indicative of similar signals x_1 and x_2 . However, it should also be clear that terms in the “cor_vector,” that are due to out of phase comparison of x_1 and x_2 will yield low correlation values using this scheme. Signals which are synchronized (in phase), only require one pass through Step 3, since that will automatically yield the largest

correlation value. Signal synchronization then reduces the complexity of the algorithm to $O(N)$. Furthermore, a robust methodology is available by converting to frequency domain analysis of the signals.

Utilizing frequency analysis of the signals offers a much more convenient and valid way to compare and correlate signals in vibration and modal analysis. The common technique for this is using the Fast Fourier Transform (FFT) to convert to the complex frequency domain, then determine the power spectral density, which measures the energy at various frequencies. The formula for computing the FFT, in this case the discrete FFT (DFT), is defined:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N}nk} ; \text{ with } k = 0, \dots, N-1 \quad \text{Eq. 66}$$

Direct evaluation of these terms has a computational complexity of $O(N^2)$, but a number of algorithms have been developed that reduce the requirement to $O(N \cdot \log N)$ (Cormen, et.al., 1995) and (Ifeachor, et.al, 2002). Additionally, the power spectral density, P_X , brings the signal's FFT into the real domain by multiplying the complex FFT vector by its complex conjugate:

$$P_{X_k} = \frac{X_k \cdot X_k^*}{N} ; \text{ with } k = 0, \dots, N-1 \quad \text{Eq. 67}$$

Plotting the power spectral density versus frequency allows for two signals to be directly compared, since they have a common basis. Figure 63 shows the energy output of the accelerometer sensor versus frequency for the model with the A Constraint, while Figure 64 illustrates the energy output of the sensor for the other configuration of the model with constraints at both A and B. As above, the constraints are defined as in Figure 50. Outputs from both models are plotted together for comparison in Figure 65. As can be seen, having the signals represented in the frequency domain allows for the signal differences to be easily seen and quantified.

Correlation of the power spectral density functions gives a means of quantifying the similarities of the signals. In this case, applying the correlation algorithm detailed above yields the following correlation matrix, C:

$$C = \begin{bmatrix} C_{A,A} & C_{A,AB} \\ C_{AB,A} & C_{AB,AB} \end{bmatrix} = \begin{bmatrix} 1 & 0.24 \\ 0.24 & 1 \end{bmatrix} \quad \text{Eq. 68}$$

$C_{A,A}$ and $C_{AB,AB}$ represents the autocorrelation with each signal by itself, with A being the A constrained model, and AB the A and B constrained model configuration. Since they are the same signal, their correlation should be 1. Here though, the diagonal terms are of

primary significance, these represent the cross correlation between the A constraint model, and the AB constraint model. This shows a correlation ratio of 24% between the two signals. The cross correlation ratio could be used to classify the different signals similar to the illustration in Figure 49.

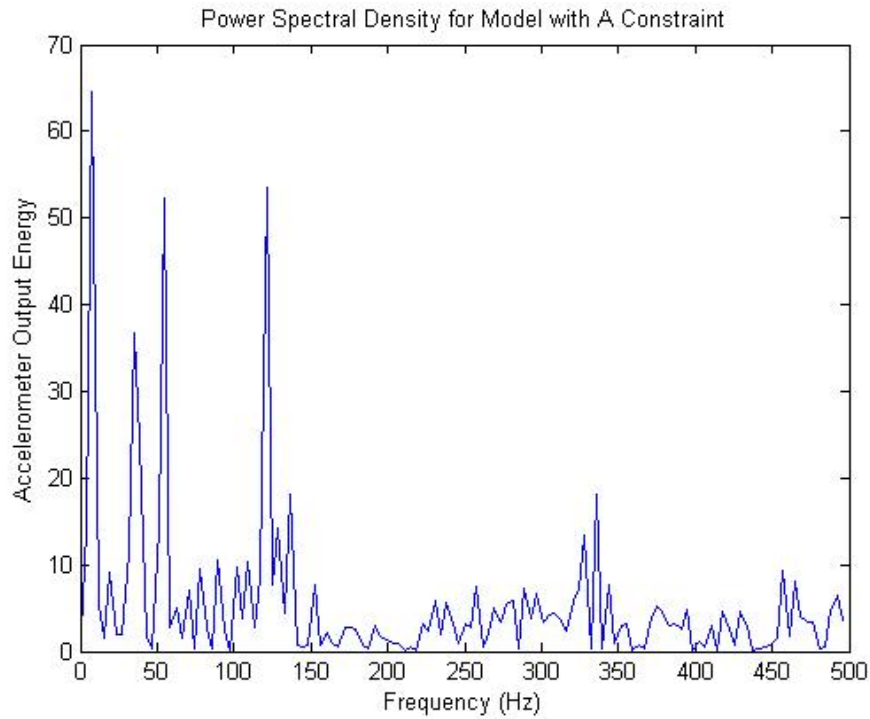


Figure 63: Frequency Domain for Model with A Constraint

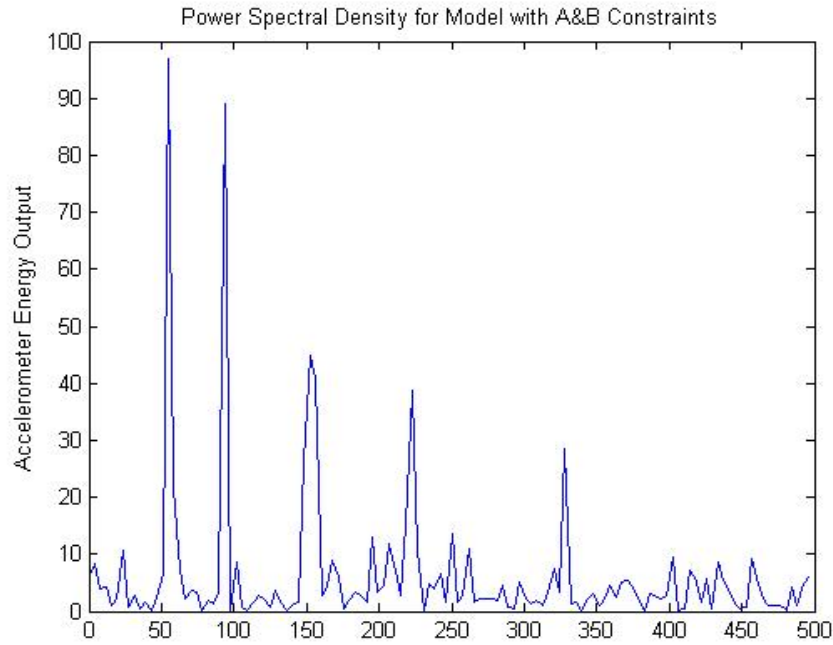


Figure 64: Frequency Domain for Model with A and B Constraints

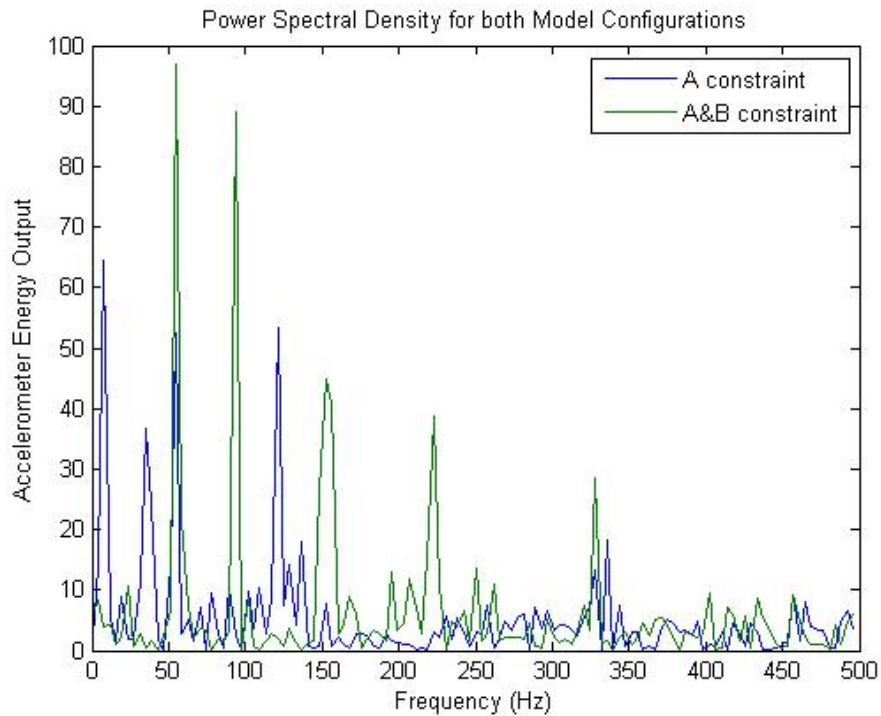


Figure 65: Common Basis for Comparison of Both Model Configurations

Results

Two FEA models; 1) a PCI circuit card, and 2) an example Rotor-Hub, were developed here to generate accurate modal shapes and natural vibration frequencies typical of existing “real-world” systems. Accelerometer output, simulated with noisy signals characteristic of the system natural frequencies were generated for each Model in its NORMAL state, and in various FAILED states. Signal processing algorithms including, FFT, power spectral density, and cross correlation were used to classify the different model signals. The results are presented here to illustrate the capability of the methodologies presented.

Initially, the systems were simulated under typical operating conditions, with no faults to simulate the “NORMAL” state. Mode shapes and natural frequencies associated with the NORMAL state were generated. Then, faults characteristic to the device were introduced to the FEA models. New mode shapes and natural frequencies, characteristic of the “FAILED” component state, were generated. The inserted faults were designed to simulate physical degradation of system components that commonly exist in current military systems. As detailed above, noisy signals representative of sensor outputs from modal vibrations are simulated for each model configuration represented.

PCI Card Simulation

The first model simulated is an accurate representation of a standard form factor PCI bus circuit card as shown in Figure 66. The card is constrained at four locations, A – D. Points A and B represent riveted, or screw attachments to a metal bracket, which is used to secure the card to a cabinet or chassis. Common failure modes at those points could be shear of the rivets and/or delamination of the board at the rivet points. Both failure types would induce loss of constraint. Areas C and D model the compression connections to the PCI bus. Common failure modes in these areas are cracking along the board in the longitudinal direction, resulting in loss of constraint degrees of freedom in those areas, especially in the transverse direction (in and out of the board).

The standard form factor dimensions, and the mechanical properties utilized for FR4 composite material, commonly used in circuit board fabrication are detailed in Table 6.

A stress analysis with transverse pressure loading was conducted to illustrate areas within the board subject to elevated stress. This simulates a standard load due to acceleration in the transverse direction. Areas of elevated stress are typically candidates for initiation of cracking. The PCI bus connection in Area D, and the rivet point, A, are the areas of highest von Mises stress, and likely correspond to initial failure sites. This is illustrated in Figure 67.

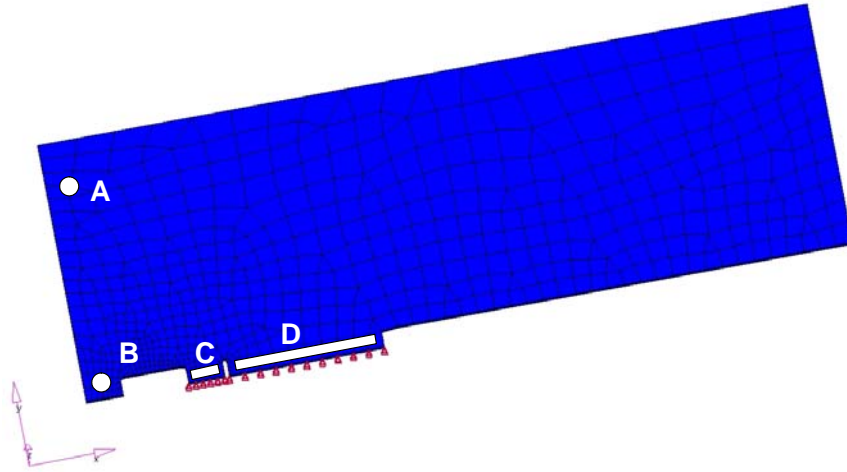


Figure 66: PCI Bus Card, Constraints at A, B, C, D

Table 6: PCI Card Mechanical Properties

Mechanical Properties of PCI Bus Card (FR4 Composite)	
Major dimensions	3.175 mm(D) x 107.0 mm(W) x 312.0 mm(L)
Density, ρ	1820 kg/m ³
Young's Modulus, E	18.62 GPa
Poisson's Ratio, ν	0.136

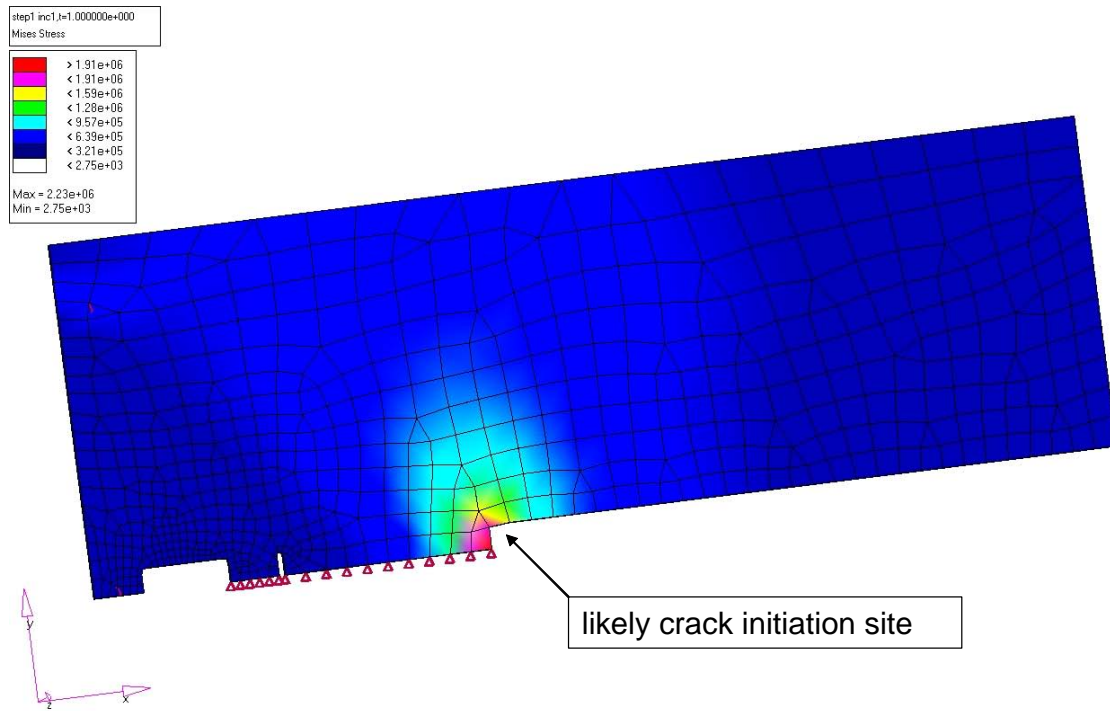


Figure 67: PCI Bus Card Subjected to Lateral (Mode 1) Vibration Loading (Von Mises Stress)

The following configuration list summarizes the candidate failure points.

PCI Card Model Configurations

- NORMAL: Normal operating conditions
- Fault A: Implies cracking in area A.
- Fault B: Implies cracking in area B.
- Fault C: Implies cracking in area C.
- Fault D: Implies cracking in high stress area D.

Each fault was simulated singly utilizing the FEA analysis software. Multiple and combination fault permutations were not considered. Table 7 summarizes the resulting natural frequencies for the first ten vibration modes associated with NORMAL system operation, and for each fault condition, A-D.

Table 8 shows the mode with maximum difference from NORMAL operation for each Fault. Note that Fault C differs little from the NORMAL operation and will be difficult to detect. Figure 68 and Figure 69 illustrate the physical difference in Mode 4 vibration for Fault A versus NORMAL operation. Figure 70 and Figure 71 illustrate the physical difference in Mode 7 vibration for Fault B versus NORMAL operation. Figure 72 and Figure 73 illustrate the physical difference in Mode 8 vibration for Fault C versus NORMAL operation. Figure 74 and Figure 75 illustrate the physical difference in Mode 1 vibration for Fault D versus NORMAL operation. The figures show the mode of maximum difference for each respective fault versus the NORMAL operation.

Table 7: PCI Card Natural Frequencies

MODE	NORMAL	Natural Frequency, ω_n (Hz)			
		Fault A	Fault B	Fault C	Fault D
1	33.1	32.876	33.074	33.101	21.147
2	132.6	103.05	131.83	132.55	119.72
3	217.2	200.83	217.11	217.17	144.55
4	361.1	218.10	358.41	360.17	338.15
5	609.6	393.68	609.03	608.47	423.48
6	666.4	612.06	663.85	663.46	558.27
7	846.3	679.83	726.68	845.36	641.29
8	951.7	730.47	842.65	926.09	766.82
9	1070.0	894.63	972.85	1066.9	979.82
10	1128.2	1024.8	1070.8	1111.2	1068.0

Table 8: Damaged PCI Card, Deviation of Natural Frequencies from Normal

Percent Deviance from Normal State				
MODE	Fault A	Fault B	Fault C	Fault D
1	-0.7%	-0.1%	0.0%	-36.1%
2	-22.3%	-0.6%	0.0%	-9.7%
3	-7.5%	0.0%	0.0%	-33.4%
4	-39.6%	-0.7%	-0.3%	-6.4%
5	-35.4%	-0.1%	-0.2%	-30.5%
6	-8.2%	-0.4%	-0.4%	-16.2%
7	-19.7%	-14.1%	-0.1%	-24.2%
8	-23.2%	-11.5%	-2.7%	-19.4%
9	-16.4%	-9.1%	-0.3%	-8.4%
10	-9.2%	-5.1%	-1.5%	-5.3%

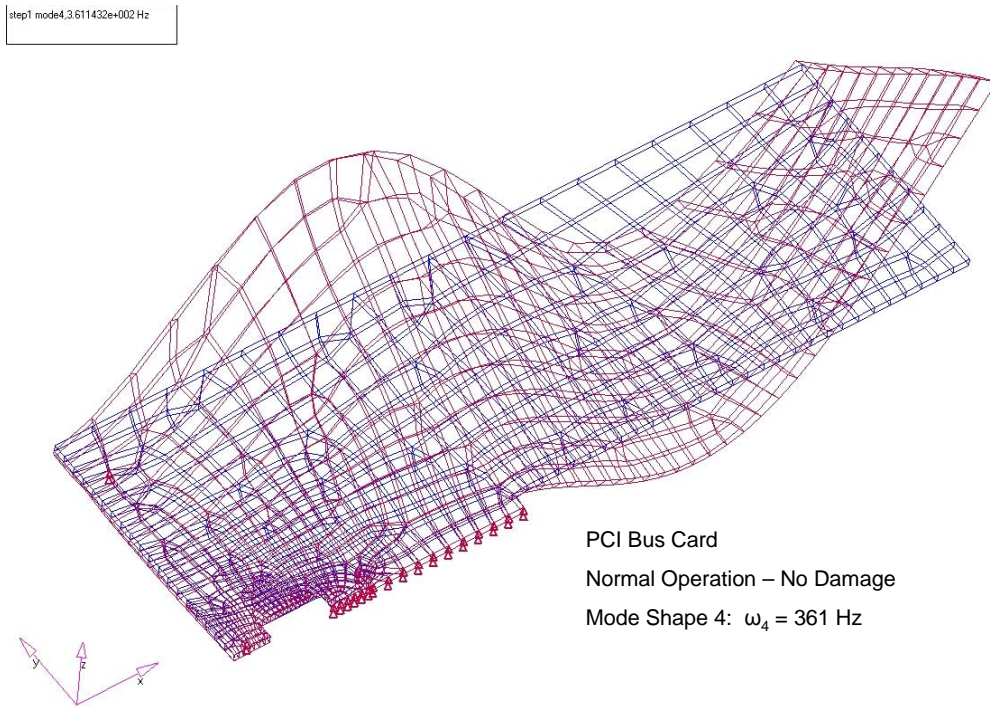


Figure 68

NORMAL vs. Fault A, Dominant Mode: 4

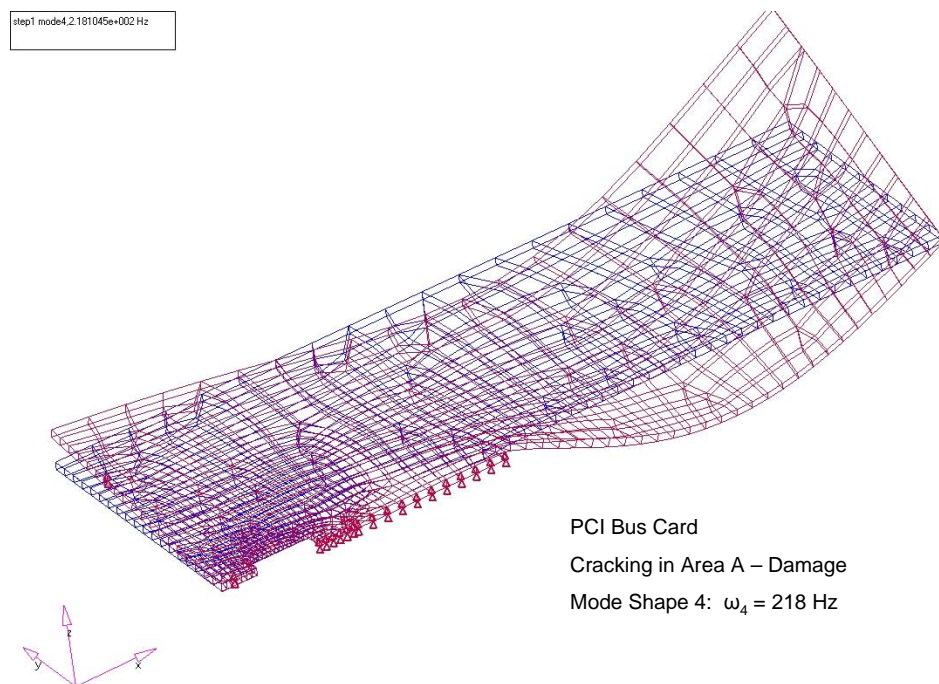


Figure 69

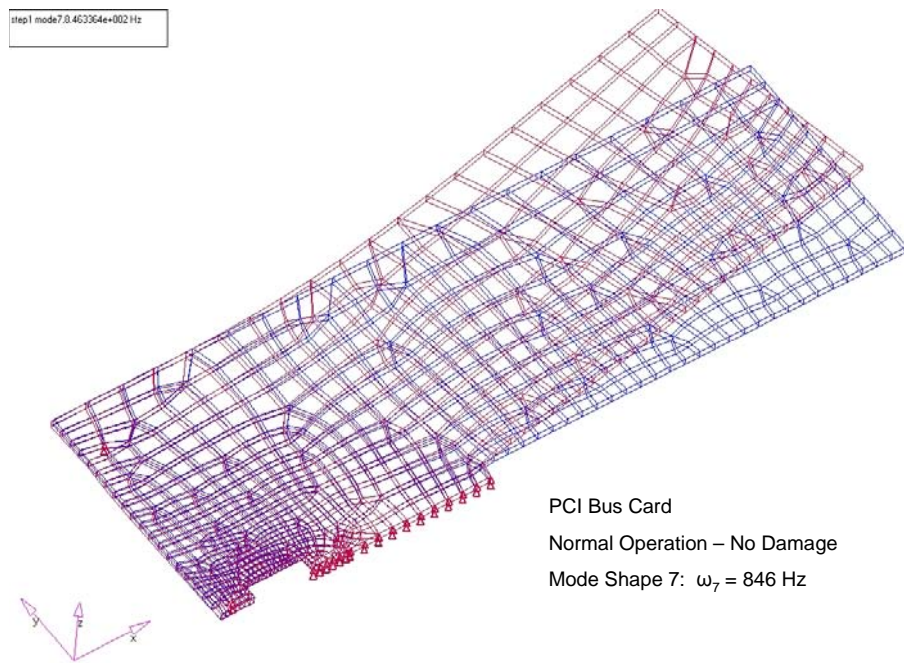


Figure 70

NORMAL vs. Fault B, Dominant Mode: 7

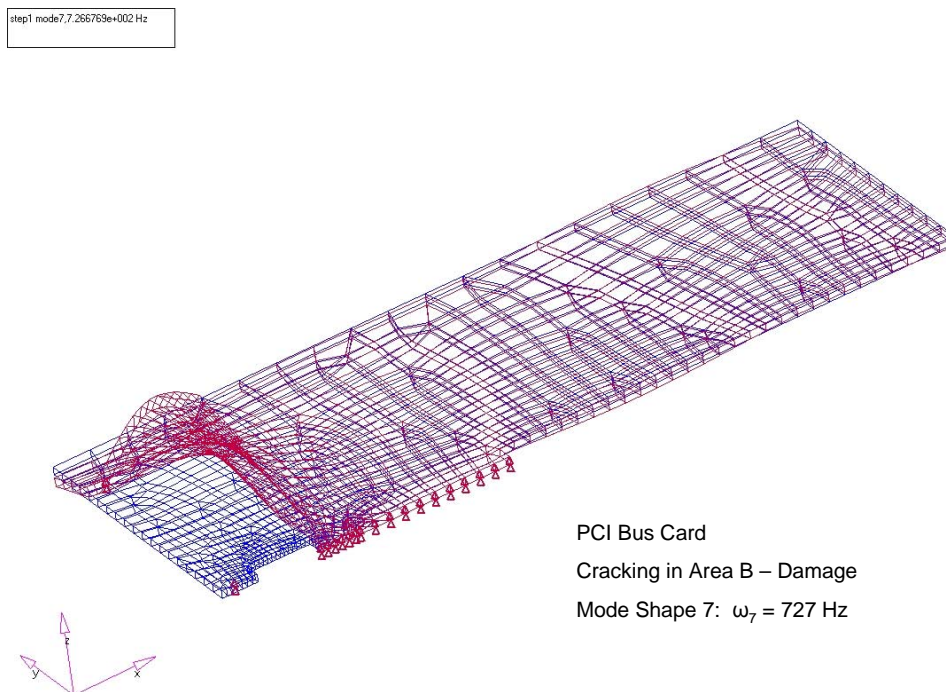


Figure 71

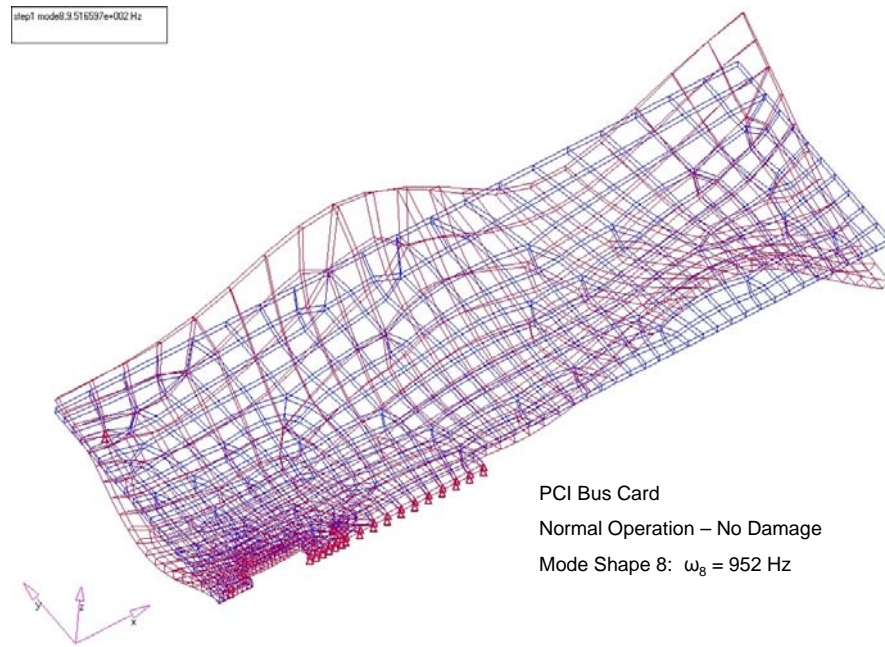


Figure 72

NORMAL vs. Fault C, Dominant Mode: 8

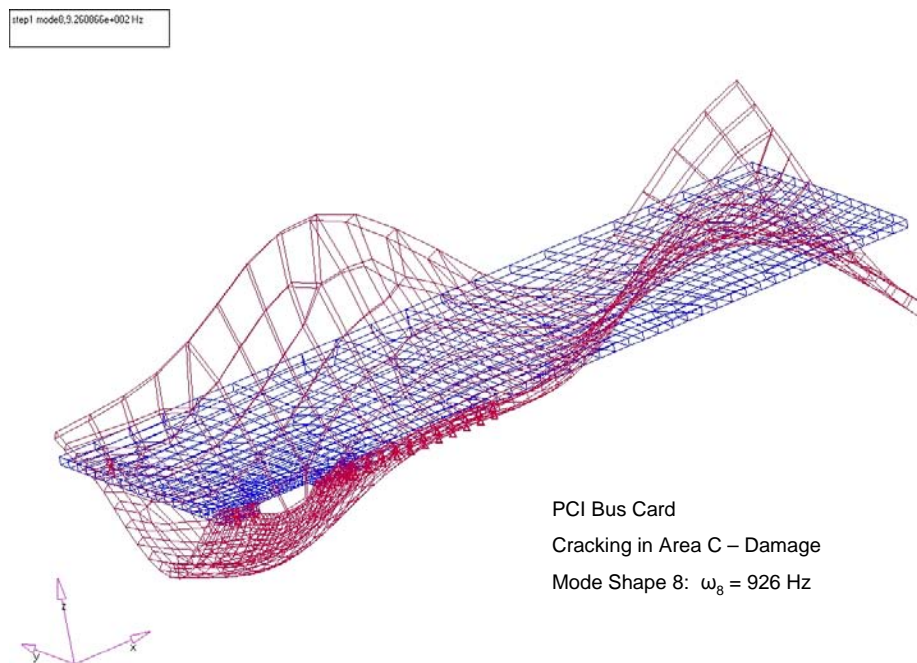


Figure 73

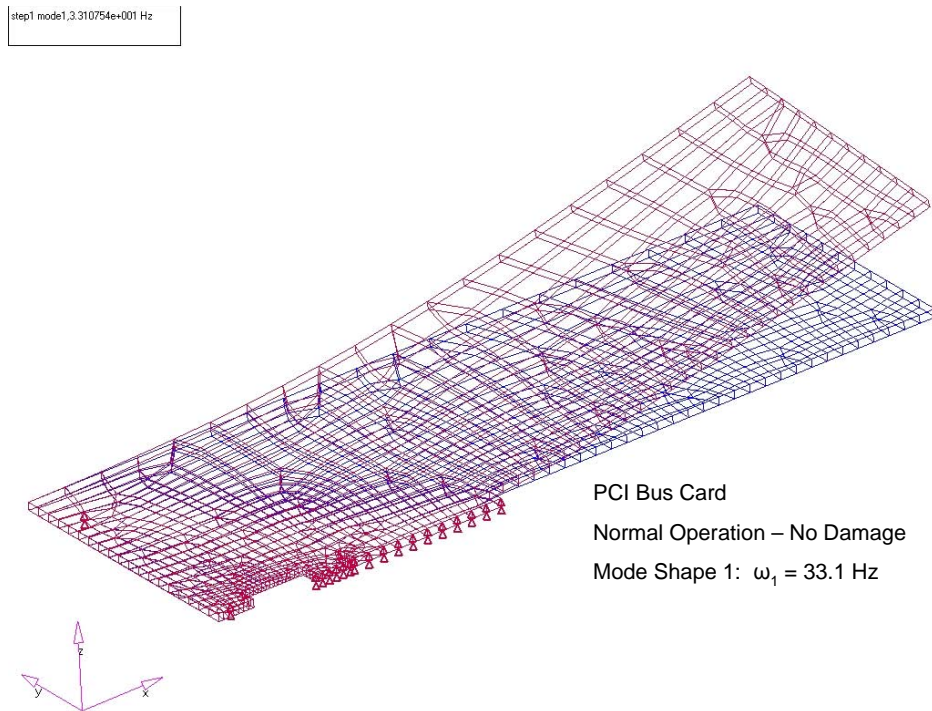


Figure 74

NORMAL vs. Fault D, Dominant Mode: 1

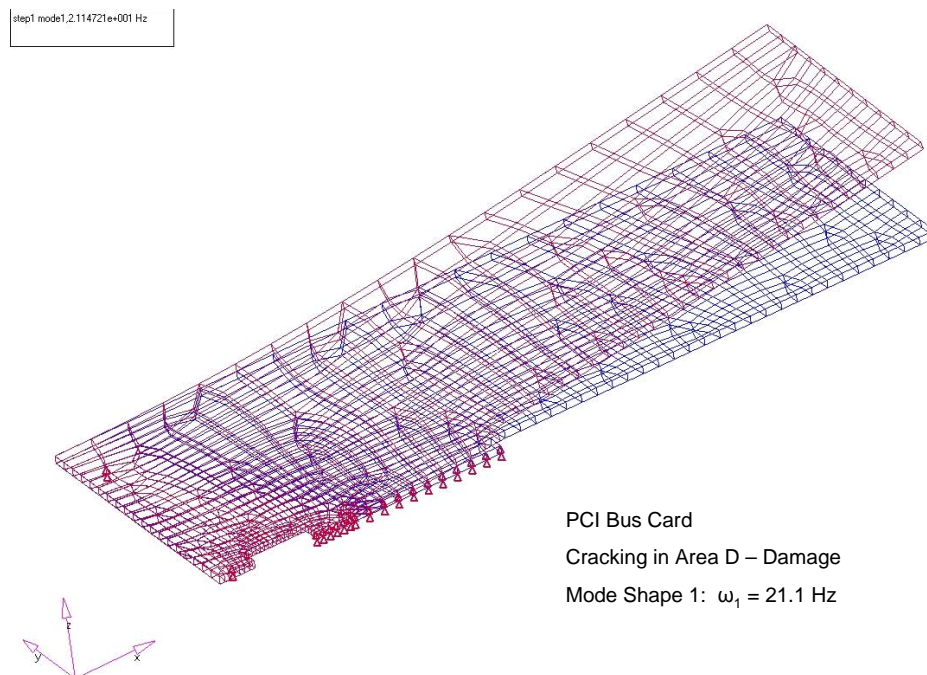


Figure 75

The simulated output sensor signals for the PCI Card models are based on the derived system natural frequencies for NORMAL operation and Faults A-D are expressed as:

$$x_k(t) = \sin(2\pi\omega_{1_k}t) + \sin(2\pi\omega_{2_k}t) + \sin(2\pi\omega_{3_k}t) + \dots + \sin(2\pi\omega_{10_k}t) \quad \text{Eq. 69}$$

for $k = [NORMAL, Fault_A, Fault_B, Fault_C, Fault_D]$

random white noise, with amplitude A is added in to the system as:

$$x_{k(noisy)} = x_{k(clean)} + A * random \quad \text{Eq. 70}$$

The digital signal is sampled at 2.5KHz, well above the Nyquist frequency. As discussed previously, the FFT (DFT) and the power spectral density of each of the k signals is calculated. Plots of the NORMAL operation vs. Fault A, Fault B, Fault C and Fault D respectively are illustrated in Figure 76, Figure 77, Figure 78 and Figure 79. Note that the power density for Fault B and Fault C look very similar to the NORMAL power density.

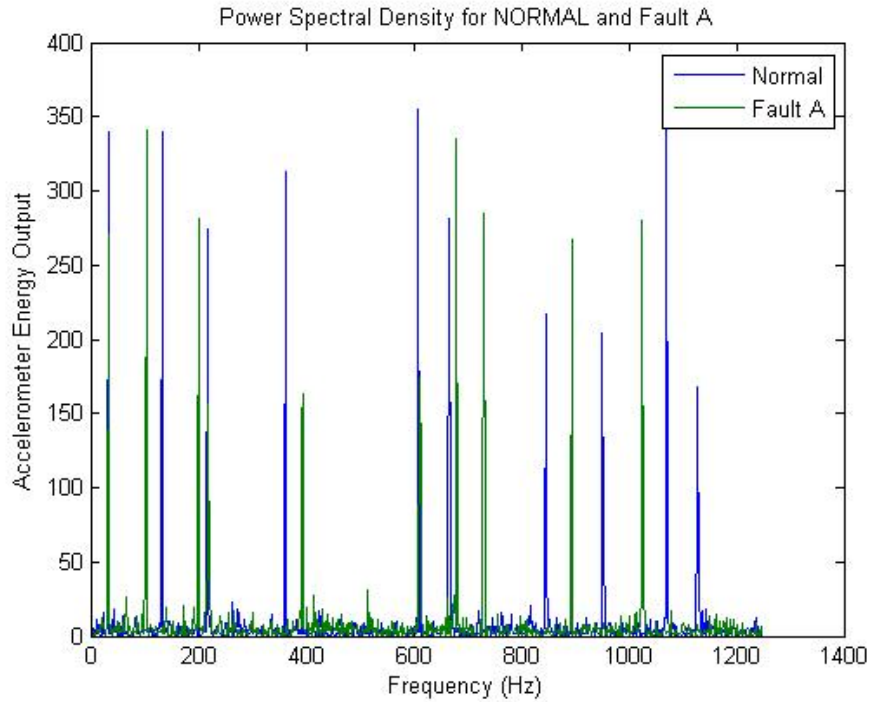


Figure 76: PCI Card, Comparison of NORMAL vs. Fault A Operation

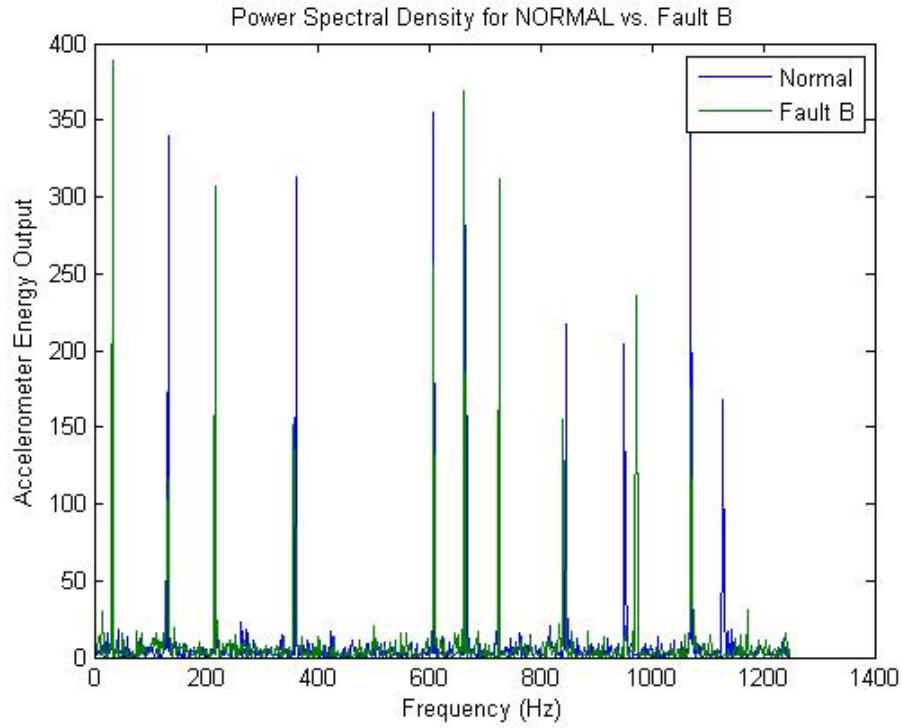


Figure 77: PCI Card, Comparison of NORMAL vs. Fault B Operation

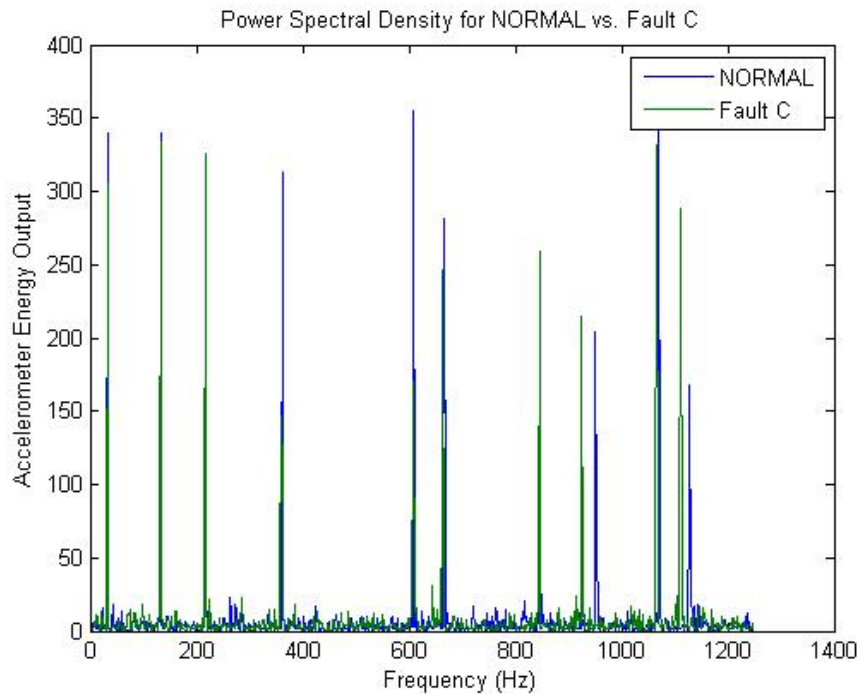


Figure 78: PCI Card, Comparison of NORMAL vs. Fault C Operation

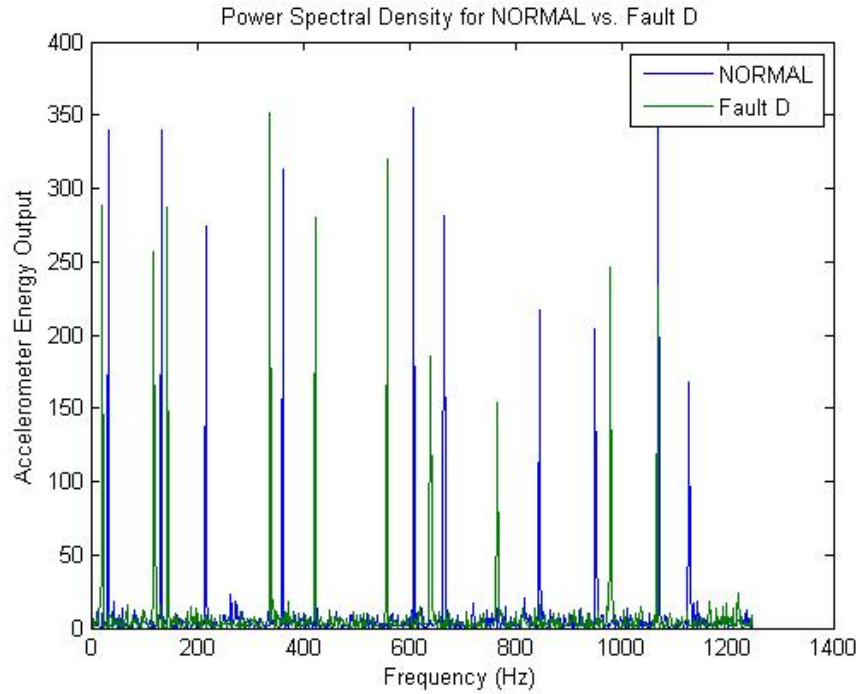


Figure 79: PCI Card, Comparison of NORMAL vs. Fault D Operation

Cross correlation of each fault signal with the NORMAL signal was performed to ascertain the extent of similarity between NORMAL and faulty operations. Results are displayed in Table 9. Recalling that low correlation values imply dissimilar states of operation, the table clearly illustrates that all four fault scenarios A-D differ significantly from the NORMAL operation. The largest difference is nearly 98% for Fault D, and the least difference is approximately 45% for Fault C. This result indicates that the methodology works very well at identifying the faulty states.

Table 9: PCI Card, Cross Correlation of Faulty to NORMAL Operation

Signal Comparison	Level of Correlation
NORMAL vs. Fault A	11.22%
NORMAL vs. Fault B	52.6%
NORMAL vs. Fault C	54.9%
NORMAL vs. Fault D	2.1%

Rotor Hub Simulation

The second model simulated is a sample representation of a rotor hub device as illustrated in Figure 80. The hub is constrained at five locations, the four bolt holes and from the top bearing area that interacts with the drive shaft. With reference to Figure 81, three potential faults are investigated. The fault condition at Point A represents a loose or worn shaft bearing condition, allowing slight movement in that region, identified as Fault A. A single loose or sheared mounting bolt fault condition is simulated at Point B, and identified as Fault B. Only a single bolt failure is modeled, multiple and combinational faults are not simulated. Fault C, illustrated in the area of C in the figure, represents a crack in the hub solid section.

The major dimensions, and the mechanical properties utilized for the steel hub material, are detailed in Table 10.

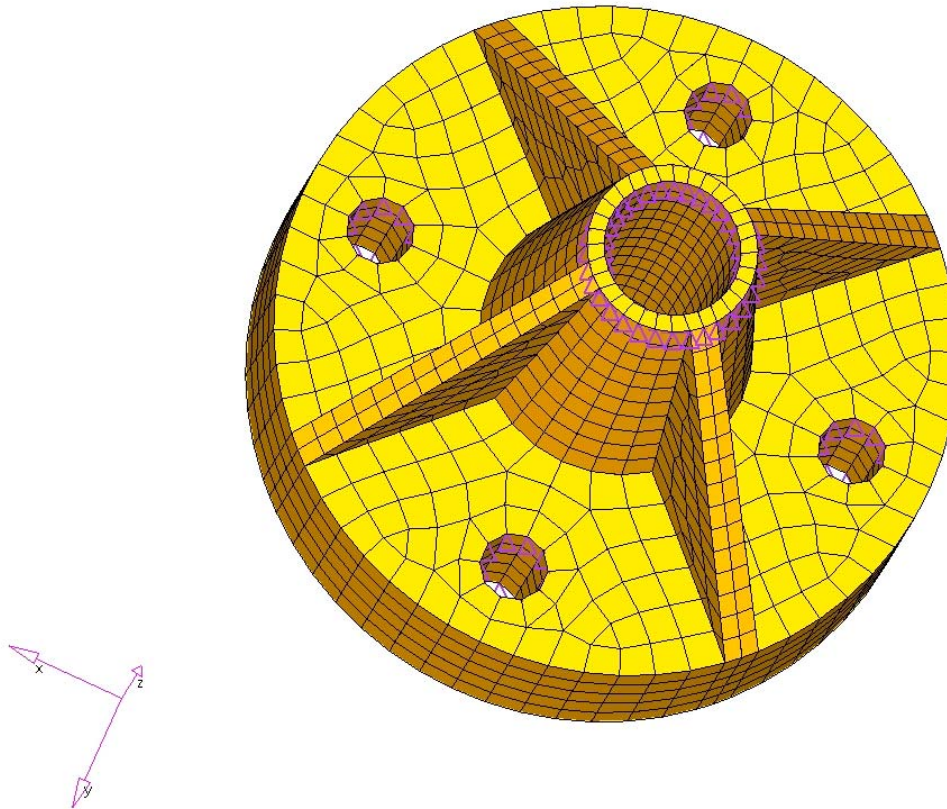


Figure 80: Simulated Rotor Hub

Table 10: Rotor Hub Mechanical Properties

Mechanical Properties of Rotor Hub (Steel) [Askeland, 1989]	
Major dimensions	10.0 cm(Radius) x 25.0 cm(H) 4.0 cm mainshaft diameter
Density, ρ	7860 kg/m ³
Young's Modulus, E	20.0 GPa
Poisson's Ratio, ν	0.300

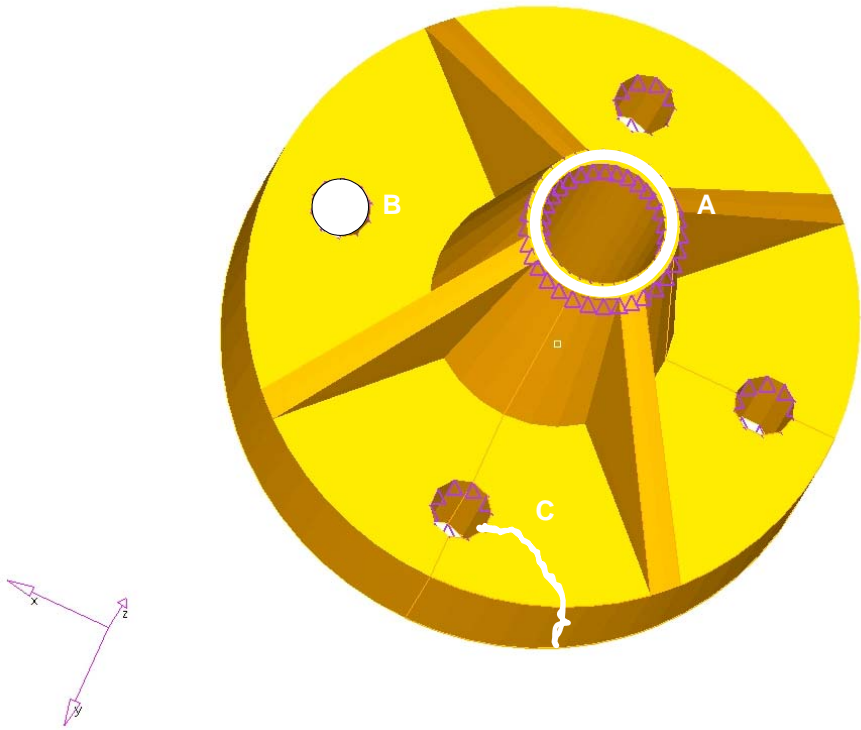


Figure 81: Rotor Hub, Potential Fault Sites

The following configuration list summarizes the candidate failure points.

Rotor Hub Model Configurations

- NORMAL: Normal operating conditions
- Fault A: Implies bearing weakness in area A.
- Fault B: Implies shearing in rotor pin area B.
- Fault C: Implies cracking in area C.

Each fault was simulated singly utilizing the FEA analysis software. Multiple and combination fault permutations were not considered. Table 11 summarizes the resulting natural frequencies for the first ten vibration modes associated with NORMAL system operation, and for each fault condition, A-C.

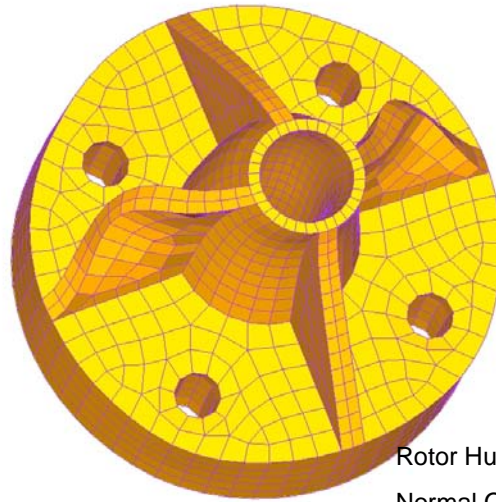
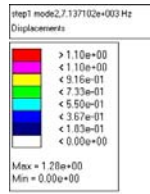
Table 12 shows the mode with maximum difference from NORMAL operation for each Fault. Note that the crack at Fault C causes only a maximum of 5% deviation in vibration mode 5 from the NORMAL operation and will be difficult to detect. Figure 82 and Figure 83 illustrate the physical difference in Mode 2 vibration for Fault A – Bearing Wear versus NORMAL operation. Figure 84 and Figure 85 illustrate the physical difference in Mode 1 vibration for Fault B – Bolt Shearing versus NORMAL operation. Figure 86 and Figure 87 illustrate the physical difference in Mode 5 vibration for Fault C – Cracking versus NORMAL operation. The figures show the mode of maximum difference for each respective fault versus the NORMAL operation.

Table 11: Rotor Hub Natural Frequencies

Natural Frequency, ω_n (Hz)				
MODE	NORMAL	Fault A	Fault B	Fault C
1	7137.1	5077.5	4469.6	6794.4
2	7137.1	5077.5	5689.2	7071.4
3	7487.4	6780.5	7046.9	7431.1
4	8414.8	6793.6	7157.4	8191.3
5	8862.7	8002.3	7764.6	8417.6
6	8922.4	8002.3	8231.2	8879.7
7	8922.4	8199.9	8414.7	8911.4
8	9087.3	8334.4	8696.6	8945.1
9	9464.0	8723.9	8898.4	9063.9
10	9464.0	8723.9	8970.1	9321.2

Table 12: Damaged Rotor Hub, Deviation of Natural Frequencies from Normal

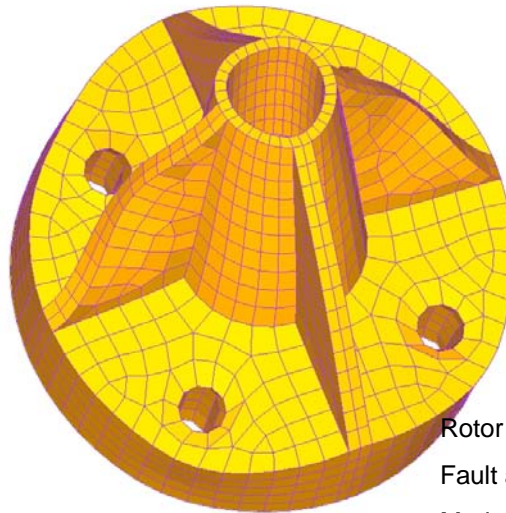
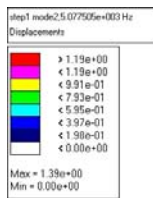
Percent Deviance from Normal State			
MODE	Fault A	Fault B	Fault C
1	-28.9%	-37.4%	-4.8%
2	-28.9%	-20.3%	-0.9%
3	-9.4%	-5.9%	-0.8%
4	-19.3%	-14.9%	-2.7%
5	-9.7%	-12.4%	-5.0%
6	-10.3%	-7.7%	-0.5%
7	-8.1%	-5.7%	-0.1%
8	-8.3%	-4.3%	-1.6%
9	-7.8%	-6.0%	-4.2%
10	-7.8%	-5.2%	-1.5%



Rotor Hub
Normal Operation
Mode Shape 2: $\omega_2 = 7137$ Hz

Figure 82

NORMAL vs. Fault A, Dominant Mode: 2



Rotor Hub
Fault at A – Bearing Damage
Mode Shape 2: $\omega_2 = 5077$ Hz

Figure 83

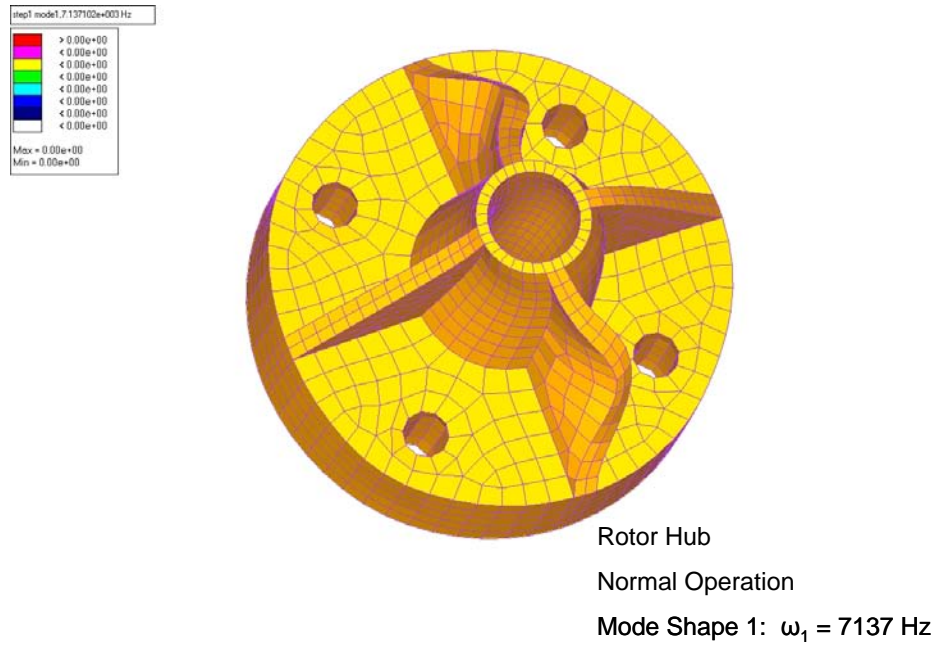


Figure 84

NORMAL vs. Fault B, Dominant Mode: 1

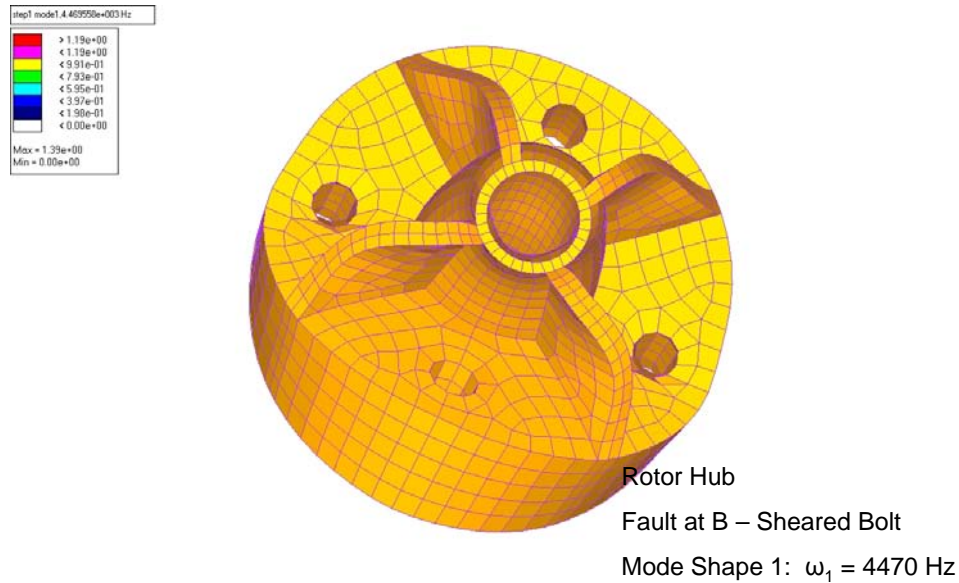


Figure 85

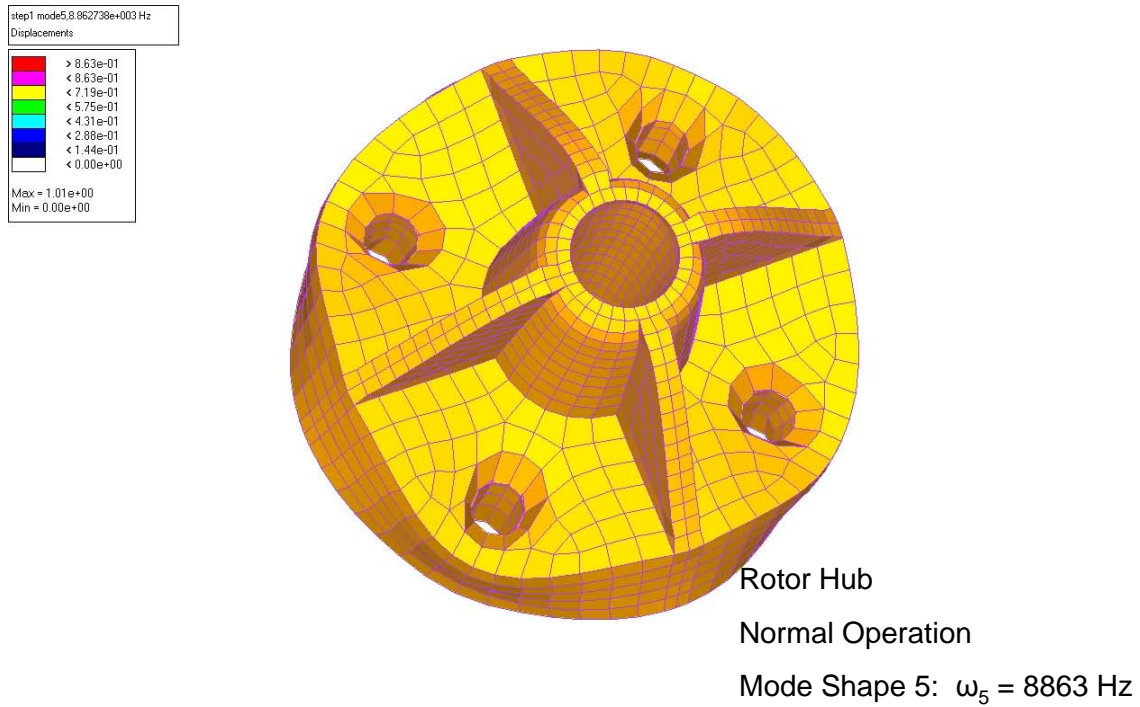


Figure 86

NORMAL vs. Fault C, Dominant Mode: 5

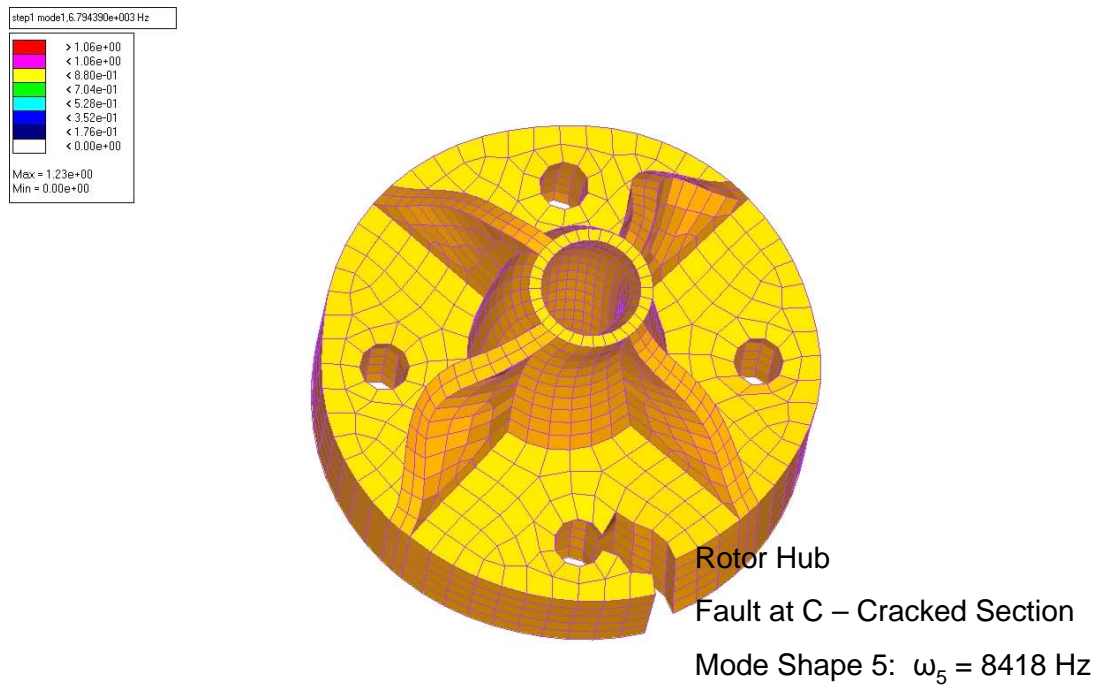


Figure 87

The simulated output sensor signals for the Rotor Hub models are based on the derived system natural frequencies for NORMAL operation and Faults A-C are expressed as:

$$x_k(t) = \sin(2\pi\omega_{1_k}t) + \sin(2\pi\omega_{2_k}t) + \sin(2\pi\omega_{3_k}t) + \dots + \sin(2\pi\omega_{10_k}t) \quad \text{Eq. 71}$$

for $k = [NORMAL, Fault_A, Fault_B, Fault_C]$

random white noise, with amplitude A is added in to the system as:

$$x_{k(noisy)} = x_{k(clean)} + A * random \quad \text{Eq. 72}$$

The digital signal is sampled at 20.0 kHz, just above the Nyquist frequency. As discussed previously, the FFT (DFT) and the power spectral density of each of the k signals is calculated. Plots of the NORMAL operation vs. Fault A, Fault B and Fault C respectively are illustrated in Figure 88, Figure 89 and Figure 90. Note that the power density for Fault B and Fault C look very similar to the NORMAL power density.

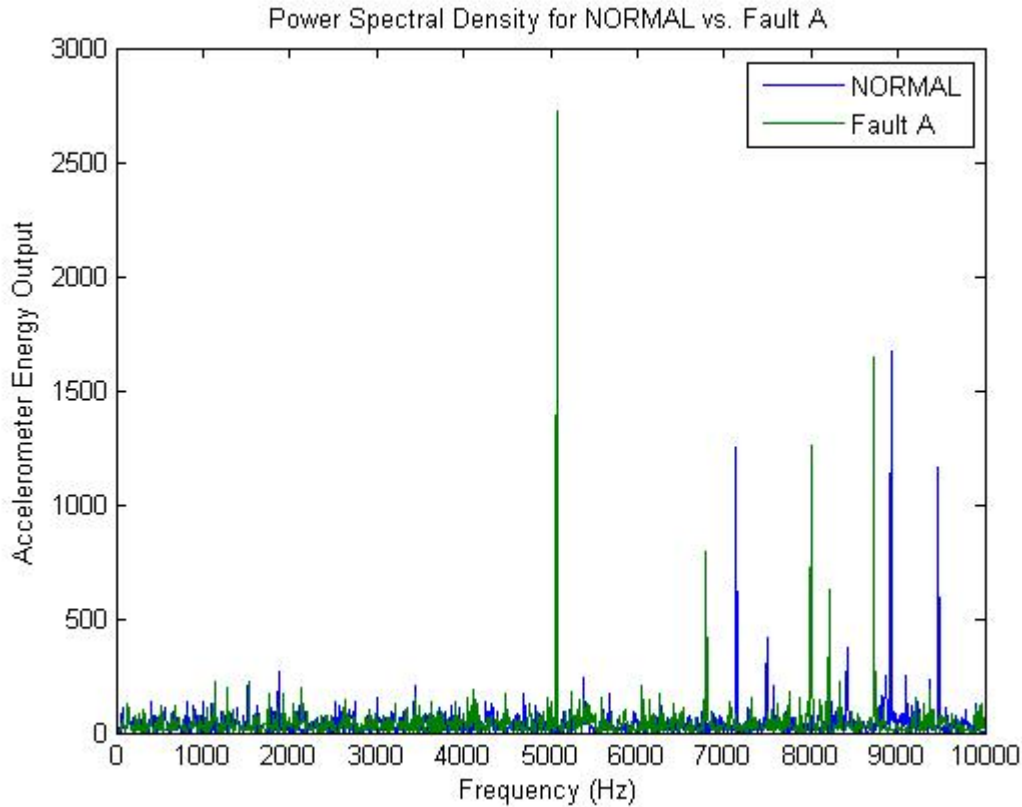


Figure 88: Rotor Hub, Comparison of NORMAL vs. Fault A (Bearing Wear)

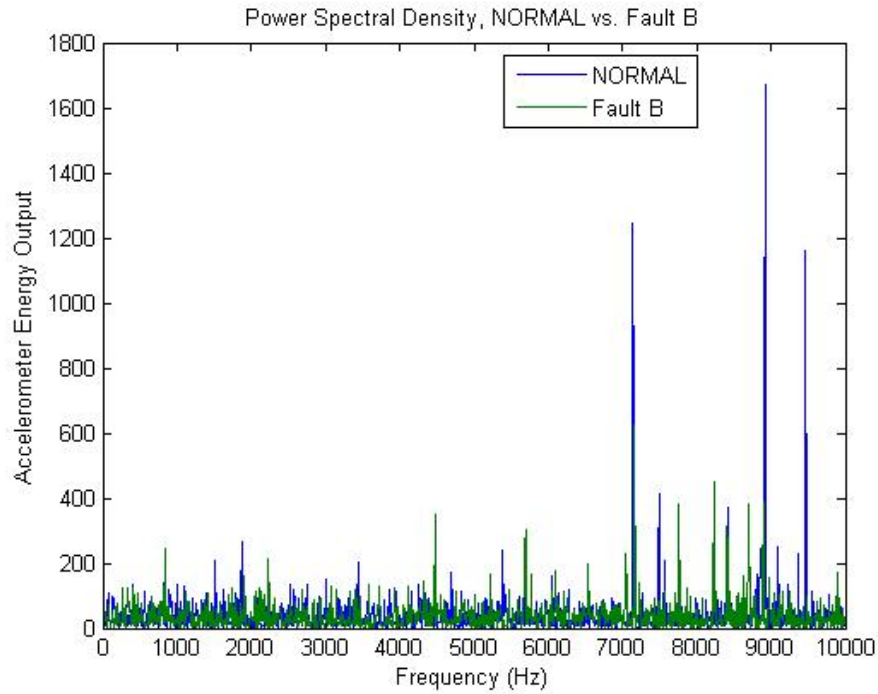


Figure 89: Rotor Hub, Comparison of NORMAL vs. Fault B (Sheared Mounting Bolt)

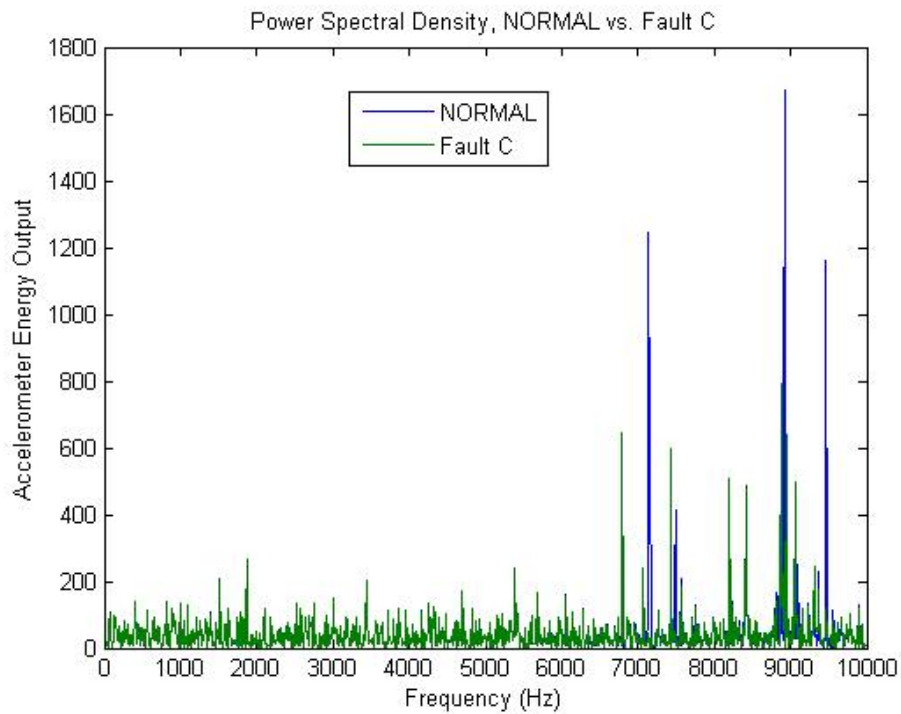


Figure 90: Rotor Hub, Comparison of NORMAL vs. Fault C (Cracked Section)

Cross correlation of each fault signal with the NORMAL signal was performed to ascertain the extent of similarity between NORMAL and faulty operations. Results are displayed in Table 13. Recalling that low correlation values imply dissimilar states of operation, the table clearly illustrates that all four fault scenarios A-C differ significantly from the NORMAL operation. The largest difference is more than 99% for Fault B (sheared mounting bolt), and the least difference is approximately 66% for Fault C (cracked section). This result indicates that the methodology works very well at identifying the faulty states.

Table 13: Rotor Hub, Cross Correlation of Faulty to NORMAL Operation

Signal Comparison	Level of Correlation
NORMAL vs. Fault A	1.56%
NORMAL vs. Fault B	0.89%
NORMAL vs. Fault C	34.0%

Pattern Recognition for Dynamic Cracking

To simulate a dynamic cracking environment, a stress analysis with transverse pressure loading was conducted to determine areas within the board subject to elevated stress, as shown in Fig. 91. The pressure load is applied perpendicular to the board face, and represents the load geometry for the maximum device response. This simulates a standard load due to acceleration in the transverse direction. Areas of elevated stress are typically candidates for initiation of cracking. The PCI bus connection in Area D is the site of highest von Mises stress (maximum distortion strain energy), which corresponds to the most likely location for crack initiation under the pressure load.

The PCI card was then simulated under progressively deteriorating ($0.0\text{m} \leq \text{crack_length} \leq 0.0695\text{m}$) cracking conditions in Area D. A single crack was simulated, beginning as shown in Figure 79, and progressed parallel to the bus junction. Simulations were conducted at eleven intervals along the crack length. The crack was introduced to the model via relaxation of the nodal constraints along the crack path. Results were combined in Fig. 92, which illustrates the resulting change in natural frequencies (Hz) for the first ten vibration modes as a function of crack length. Note that all modes show a downward trend with increasing crack length, particularly Modes 5, 7 and 8.

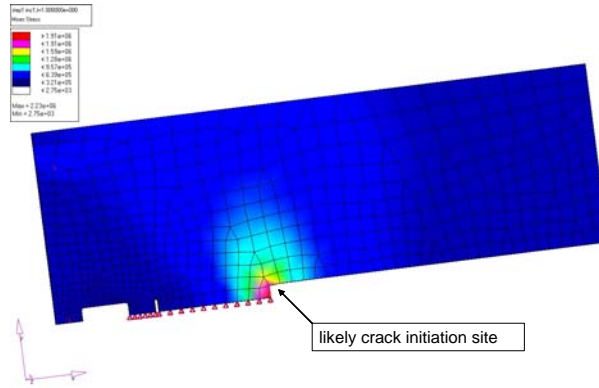


Fig. 91: Von Mises Stress due to Lateral Load

Analysis of these shifts via the signal correlation techniques described above enables detection of crack initiation and progression in a component. **Fig. 93** illustrates the level of correlation of the normal operation signatures to the signature of crack length in Area D. Note that a crack length of zero (no deterioration) corresponds to a correlation of 1.0, denoting no deviation from normal signature, indicating a healthy card. As the crack length grows, the level of correlation to the normal signal decreases, providing an indication that the condition, or health, of the circuit card is deteriorating.

It is worth noting here that higher fidelity finite element models will enable detection of smaller crack lengths. In this simulation, the increment of crack growth is equal to the distance between adjacent nodes on the crack path. Increasing the nodal density should create a smoother version of the data from **Fig. 93**, but comes at the expense of increasing the computational effort of the simulation.

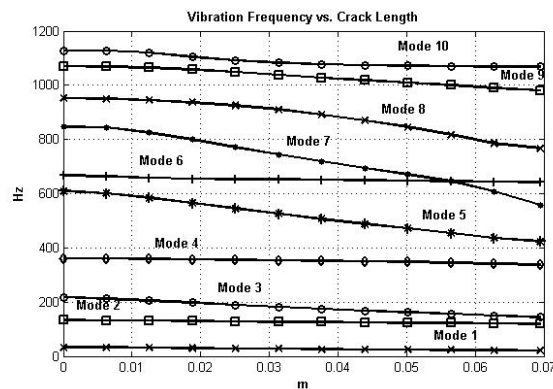


Fig. 92: Modal Frequency as a Function of Crack Length

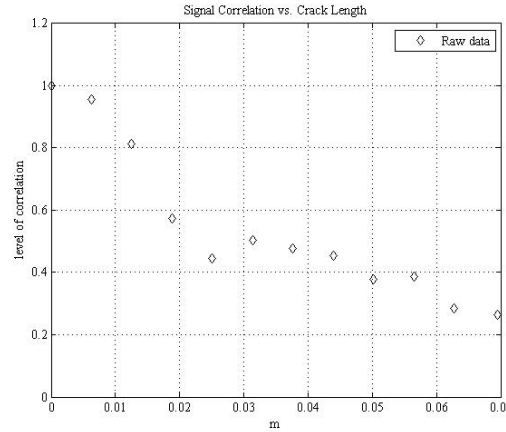


Fig. 93: Level of correlation of normal signal to deteriorating component signal as a function of crack length.

Conclusions

A theoretical framework and algorithmic methodology for obtaining useful diagnostic and prognostic data from electro-mechanical systems was developed and presented. The methods were based on vibration and modal analyses of the physical components. Two “real world” models, a PCI circuit card, and an example rotor hub were simulated. Application of the developed techniques proved very useful in identifying FAULTS on the simulated systems.

The physical models were created using finite element analysis (FEA). These models were simulated to obtain the first ten mode shapes, and concomitant natural vibration frequencies of the unadulterated models. Faults characteristic of the modeled components were introduced to each FEA simulation. For the PCI card, four different faults, two loose mounting rivets and two longitudinal cracks were introduced to the original model. For the rotor hub, fault conditions representative of a worn bearing, a sheared bolt, and a section crack were impressed on the original model. The new mode shapes and natural frequencies due to the faults were obtained for each fault singly.

The obtained system natural frequencies for each NORMAL and FAULT states were used to simulate output from an accelerometer. Significant white noise was introduced to the signal to make the simulation more realistic. To provide a common basis for comparison of the signals, they were transformed to the frequency domain using the well-known digital Fourier Transform algorithm, FFT. For correlation, the Power Spectral Density was computed for each signal. A signal processing algorithm called Cross Correlation was performed to compare each FAULT test signal vector to the NORMAL benchmark signal.

A high correlation value (100% is highest) between the test and benchmark signals indicates a good match to the NORMAL benchmark signal, which implies that the test

signal is also NORMAL. Conversely, a low correlation value between the test and benchmark signals is indicative of a poor match to the NORMAL signal, which could imply an IMMINENT or near FAILURE. The test signals for all faults introduced to the simulated PCI board and rotor hubs, produced distinctly low correlation to NORMAL operation, ranging from a low of 0.89% for the sheared bolt on the rotor hub, to a high of 55% for a small crack on the PCI card.

The results obtained from these simulations are very encouraging, and indicate that using modal analysis and frequency domain vector correlation may be very useful in characterizing overall system health.

This work investigated the feasibility of using pattern recognition techniques to monitor small devices that are subject to frequent sustained excitation from active external sources in real-time, and whether small differences in material health could indeed be detected with low algorithmic complexity. A theoretical framework and algorithmic methodology for obtaining useful diagnostic and prognostic data was developed and presented. The methods were based on vibration and modal analyses of the physical components. A standard PCI circuit card was simulated to demonstrate the methodology. Application of the developed techniques proved very useful in identifying faulty and deteriorating conditions on the simulated system.

The physical models were created using finite element analysis (FEA). Faults characteristic of the modeled components were introduced to each FEA simulation. For the PCI card, four different faults, two loose mounting rivets and two longitudinal cracks were introduced to the original model. Furthermore, a dynamic longitudinal crack initiating at the highest stress point on the circuit card was introduced to simulate deteriorating conditions. Each state was simulated to obtain results for comparison to the normal results. The first ten mode shapes, and concomitant natural vibration frequencies were obtained for each condition.

The obtained system natural frequencies for each state were used to simulate output from an accelerometer. To provide a common basis for comparison of the signals, they were transformed to the frequency domain using the well-known digital Fourier Transform algorithm, FFT. For correlation, the Power Spectral Density was computed for each signal. A cross correlation signal processing algorithm was utilized to compare each faulty and deteriorating test signal vector to the normal benchmark signal. The test signals for all faults, and deteriorating state cracks introduced to the simulated PCI board produced low correlation to normal operation. The strength of correlation was strongly related to crack length in the deteriorating states.

It is worth noting that the changes in the power spectral density are not independent of the crack direction. The mode shapes and frequencies are driven by the geometry of the device. Therefore, changes in crack direction strongly affect the particular mode shapes and concomitant frequencies that are dominant in the crack direction. That is why multiple modalities should be examined (the first 10 in this case). While not investigated here, this characteristic can help classify not just faulty operation, but aid in identifying

the specific type of fault that is in progress, since physical devices typically fail in a finite number of known modalities. Conceivably, each of these could be classified according to a unique modal signature. The results obtained from these simulations are encouraging, and indicate that using modal analysis and frequency domain vector correlation is a promising technique for use in characterizing overall system health.

Student Project 1: SHELPP – Data Mining of ATS and UUT Information

This section details work on a software project dubbed SHELPP, which was expanded upon by students under this project. Students involved with this project include:

Mr. Obinna Odumodu, EE Graduate
Mr. Babatunde Williams, EE Undergraduate
Mr. James Seals, EE Undergraduate
Mr. Gerald Saltus, EE Undergraduate
Mr. Coley Coleman, CS Graduate

Background

Bayesian methodologies can be used for adaptive learning in embedded diagnostic systems. In this framework, the overall aim is to develop probabilistic models that are well matched to the data, and make optimal predictions with those models. With Bayesian techniques, the probability of a specific outcome in individual relation to an event is provided by its past history of occurrences. With this information, a network can be built that analyzes the effects one event has on another, and how these effects propagate into other systems. This analysis is pure probability and because of that, a Bayesian network will not completely characterize a system's working, but possibly it will trend precursors to catastrophic failures. This of course would lead to improvements in things such as the rate of failure based upon peripheral events like the coupling effects with connecting systems. Essentially this will also tie to the premonition of better quality and reduced downtime if any. Decomposition is the key to this task and is in large something that Bayesian networks require.

There may even be relationships of the upper limit, lower limit, and measured value of different test steps. For example, the measured value of a test step has a nominal, or usual, value. Let us keep in mind that a test program is made to stop on first test failure so there isn't a valid way to relate a failed test to subsequent test. Though, the measured value of a test (s) previous to the failed test step may, according to their difference from their nominal value if their nominal values are different, be indicators for the failed test. This idea brings up the concept of permutations and combinations.

Bayesian Modeling Software Evaluation:

A survey of available modeling software was conducted to determine which were most suitable for the development of our product. The initial choice was GeNIe (Graphical Network Interface) software package which could be used to create decision theoretic models intuitively using the graphical click-and-drop interface. GeNIe is the graphical interface to SMILE (Structural Modeling, Inference, and Learning Engine), a fully portable Bayesian inference engine developed by the Decision Systems Laboratory and thoroughly tested in the field since 1998. The click and drop interface seemed most popular with most modeling software available, however it did require the user to have

some prior knowledge. In contrast there was a certain tool which used a different approach to building models. (<http://www2.sis.pitt.edu/~genie/about.html#smile>) The WinMine Toolkit is a set of tools that allow you to build statistical models from data. WinMine was developed by the Machine Learning and Applied Statistics group of Microsoft Research and it can be freely downloaded from the Web site: (<http://research.microsoft.com/~dmax/WinMine/ContactInfo.html>)

The majority of the tools are command-line executables that can be run in scripts. WinMine has the ability to convert raw data into the XML data format by running the interactive conversion wizard DataConverter.exe and intern created a dependency network or model representation of associated variables. In essence we attempted trials with both modeling methods to derive which would be most easily interfaced with the current ATE systems. After evaluating their functionality we felt it best to continue our efforts with the WinMine software. It seemed to coincide more with our goal, because it creates feasible relationships, as opposed to the user's creation of these relationships.

Approach:

As previously stated WinMine has the ability to convert raw data into the WinMine XML data format by running the interactive conversion wizard DataConverter.exe. In the example two files were given, a transactions.raw and a demos.raw (demographics) files. The following will explain all components involved as well as the introductory steps taken in the creation of Bayesian models using this tool. These files were extracted to our machine once the WINMINE Toolkit had been downloaded:

WinMine.html	Introduction web page
EULA.rtf	End User License Agreement
Tutorial/demos.raw and Tutorial/transations.raw	Raw data files to be used with the tutorial
Bin/DataConverter.exe	Interactive tool that converts from a raw data file or an SQL table into the WinMine XML data format
Bin/DataJoin.exe	Command-line tool that performs a join on two data files
Bin/DataSplit.exe	Command-line tool that splits a data file into a training data file and testing data file
Bin/DataCheck.exe	Command-line tool that scans a data file and outputs summary statistics
Bin/PlanEditor.exe	Interactive tool for creating and editing plan files
Bin/Dnet.exe	Command-line tool that builds a dependency network or a Bayesian network from data
Bin/DnetBrowser.exe	Interactive tool for viewing dependency networks and Bayesian networks
Bin/DnetLogscore.exe	Command-line tool that computes the log predictive accuracy of a model

Table 14: File Descriptions

It is our effort to incorporate prognostic capability or predictive capability not only into automated test equipment, but at any maintenance level. In addition we have been provided with very useful Avenger data as a test-bed. We currently maintain weekly meetings including both team and AAMU academia to converse ideas and troubleshoot needed items. Our near advancements include a useable data format for extracted data from the various CEE/BSTF as well as the RMS. We are also evaluating the idea of converting data using XML tags. This will aid largely in the transfer of data, because only needed data will be transferred aiding in the speed of the analysis process. This data extraction will go far in incorporating algorithms and other analysis tools.

A software tool has been created that statistically analyzes data to create relationship models that will signify conditionals and other otherwise unknowns to better capture or isolate faults and eventually predict faults before they occur. This tool was created to use at any level of maintenance and also for logistical planning. This could reduce downtime and improve mission readiness increase supply inventory.

Software interface: System Health by Enabling Learning and Prognostics(S-HELP):

This software tool can take user input to form a specific database query. Also it makes a relationship of the queried information. The relationship is displayed to the user for diagnostic aid and potential prognostic capability or decision support.

Relationship potential: Information obtained in a data warehouse is only as useful as one make it. After it is collected and stored, something must be done to it, or otherwise it maintains only potential value. There could be some otherwise un-noted relationship between two or three different data types. WINMINE helps achieve this data correlation and association.

Prognostics potential:

When a relationship is established between two pieces of data, one can make a model from it. For example, if x fails a lot and y fails most of the time that x fails then one may conclude that the event of x failing has some effect on the event of y failing, or vice versa, or one could model the two events probabilistically. The condition of one may indicate the condition of the other one. Data reasoning, data correlation, and data association are very important factors for creating prognostic algorithms. The more data quantity and types that become available the more genetic a fault or failure may appear relative to the data.

User requirements for S-HELP:

The project required Data assessment of historical test logs from military platforms. Owing to the possible extensiveness of data, software was required to retrieve, relate, analyze, and report the statistics of the data queried through user input. Since one of the target users is the frontline soldier, this tool was made to interact with the user in a stress free manner. The software development Kit used was Microsoft Visual Studios .Net, and the common language was Visual Basic.Net(VB.NET). VB.NET was used to merge and drive all add-in software and .Net Library objects to produce the S-HELP package.

Technical Information

One of the VS.Net library objects used was the OLE DB object which stands for Object Linking and Embedding. It provides connection, manipulation and data transportation capability for most databases. The queries and non-query commands used with the OLE DB object are written in SQL (Script Query Language). SQL is the backbone of most databases, and it is the most popular language for communicating with databases programmatically. So, using the Visual Basic .Net and OLE DB (Object Linking and embedding), a connection was established with the MS Access database, and an SQL command was setup to retrieve the datasets required to process the user query.

The Sample Database

It was imperative that a sample database be created, since it would determine the efficiency and functionality of the S-HELP tool. Since the WINMINE toolkit takes in a raw text file with comma-delimited datasets, the database had to be structured in such a way that the raw text file could easily be generated while keeping with the actual structure of the ATMS database. The S-HELP database was modeled after the ATMS database at AMCOM as shown in Table 4, and there were some fields that were fabricated for the sake of missing data to create an ideal instance of the ATMS database.

The database also contains tables that can be updated to include new platforms, systems, subsystems, Test groups, and test numbers. Tables will be dynamically created to suit the insertion of new platforms for analysis. The database will be ever changing, and the S-HELP software was programmed to accommodate that.

SHELP Diagnostic/Prognostic Software Development Technical Report

The purpose of this section is to provide suitable information for understanding the project implementation and directions of the SHELP Software. The documentation will cover the design considerations and the source code implementation of SHELP. SHELP has been developed for local computer systems and for web servers, however, the backbone of both systems are the same. In addition, this document will discuss the future directions of the SHELP project. This document applies to the local machine version and the web based version of the SHELP software.

System Overview:

SHELP is software that enables the analysis of Weapon systems' test data, given external factors and considerations. The Information used by the SHELP software is derived from a database of test logs acquired from military test stations. This SHELP tool takes input from the user to form a specific database query that is used to relate the queried data and performs a statistical analysis, which is displayed to the user for diagnostic aid and potential prognostic capability or decision support.

Relationship potential: Information obtained in a data warehouse is only as useful as one make it. After it is collected and stored, something must be done to it, or otherwise it maintains only potential value. There could be some otherwise un-noted relationship between two or three different data types. WINMINE helps achieve this data correlation and association.

Prognostics potential: When a relationship is established between two pieces of data, one can make a model from it. For example, if x fails a lot and y fails most of the time that x fails then one may conclude that the event of x failing has some effect on the event of y failing, or vice versa, or one could model the two events probabilistically. The condition of one may indicate the condition of another. Data reasoning, data correlation, and data association are very important factors for creating prognostic algorithms. The more data quantity and types that become available the more genetic a fault or failure may appear relative to the data.

Design Considerations:

There were many factors to consider in constructing the design of SHELPP, namely

General Constraints

Assumptions and Dependencies

Database Structure and Content

User Roles

Query Structure and Bayesian Modeling

System Architecture (User interface and data backplane)

General Constraints

The Software is required to retrieve, relate, analyze, and report the statistics of the data queried through a user's selection.

Retrieve:

The software is required to connect to a data warehouse consisting of historical test log data, and transport data to and from the database.(OLEDB adapter)

Relate:

The Software is required to produce a relationship between the data fields acquired from the database.(dependency trees algorithm and clustering algorithm)

Analyze:

Upon modeling the data, the software is required to perform an analysis of the queried data, in this case, success, failure and trend analysis for suggestive maintenance.

Report:

After the three previous data processing stages, the statistical or probabilistic outcome is to be reported to the user in a comprehensible fashion.

The above constraints were points of focus, during the development of the SHELPP software; therefore more emphasis was put into these data-mining procedures than any other feature of the software.

Assumptions and Dependencies

Data Mining-

SHELP utilizes the data mining functionality of the Winmine Toolkit, which creates a graphical model that encodes probabilistic relationships among variables of interest in a given dataset. The winmine toolkit is included in the installation package, under the static path “c:\program files\shelp\server\winmine toolkit\.”

Framework-

SHELP depends on the .NET framework 1.1 libraries. In order to install the SHELP tool, the .NET framework 1.1 should be installed beforehand on the underlying computer system. The .net frame work can be downloaded for free from the Microsoft website.

Operating System-

SHELP can be installed and executed on windows NT Operating systems or later.

Database-

SHELP uses an MSAccess database as the data warehouse for the historical test log. Upon installation the static file path would be “c:\program files\shelp\server\shelp.mdb.”

Files and executables-

SHELP utilizes rewritable .xdat,.xplan, and .xmod files which are xml variations for the winmine toolkit. It also utilizes macro executable of file type .ahk, which is recompiled during SHELP runtime by a macro compiler, which resides in the static folder path “c:\program files\shelp\server.”

Database Structure

The WINMINE toolkit, the data mining engine of the SHELP tool, takes in a dataset in a comma-delimited raw text file, therefore the database had to be structured in such a way that the raw text file could easily be generated while keeping with the actual form of the ATMS database. The S-HELP database was modeled after the ATMS database at AMCOM, which consists of a flat (one dimensional) data table as shown in Table 16. Note that this is an ideal instance of the ATMS database, given that the data contents of the actual ATMS database consist of incomplete data fields. Table 15 is a description of the field elements that appear in a standard log file.

```

                                DATE: 12/04/03 TIME: 09:39
                                STATION: pic

                                UUT P/N: 13080451-029
                                NOUN  : PNVS NIGHT SENSOR ASSEMBLY
                                SER NO : 1111
                                TM   : 9-6625-476-30
                                F/G  : 0240

                                *** UUT FUNCTIONAL TEST FAILURE ***
                                1. UUT FAILED FUNCTIONAL TEST NO. 293000
                                2. ASSOCIATED DIAGNOSTIC TEST NO. IS N/A
                                3. R/R: EOM Error

                                TN  STATUS    LL      UL      MEASURED VALUE  DIM
                                -----
                                ICD SIGNATURE RESISTOR TEST DATE: 12/04/03 TIME: 09:30
                                150000  GO      4854.0000  5366.0000  5100.5552  OHMS
                                ... ..
                                ICD SAFE-TO-TURN-ON TESTS DATE: 12/04/03 TIME: 09:31
                                151000  GO      --      GT 0.100000E+02  0.200000E+36  OHMS
                                ... ..
                                293000  NOGO      Mirror commanded past limits  74  INT

```

Figure 94: Example of a UUT log file

Table 15: Data record element description

Data Element	Source	Size	Definition
Record Type	TPS	2	Defines type of record.
Record Length	TPS	2	Sets the number of (field count) elements allowed in a record.
Work Order Number	OPER	15	Identifies a specific job, including retest number. For example, a PCB to be tested by the ATE is a job. The first 6 digits are the DS unit UIC. The last 9 make up the sequence number of the work order for that DS unit.
UUT NSN	TPS	16	UUT National Stock Number.
UUT Part Number	TPS	21	UUT Assembly or Part Number.
UUT Serial Number	OPER	15	UUT Serial Number.
End Item Code (Source of UUT)	OPER	9	A code identifying the system from which the UUT was removed for test on the ATE.
Defective Part Circuit Designator	TPS	5X12	Identifies up to 5 failed components on a UUT. These are identified by the manufacturer circuit part designator (e.g., R1, U25, C4, etc.).
UUT Owning Unit UIC	OPER	6	The unit identification code of the unit turning in or owning the UUT to be tested.
Software Revision Number	TPS	2	Version number of the TPS software.
Failure Test Step	TPS	6	Test language statement number of test step within the TPS.
Failure Value	TPS	10	Readout/failure value/test limitation, at test step of failure.
Diagnostic Time	SYS	6	The time from the start of a TPS run to the completion of the run.
Date and Time	SYS	12	Date and time (local) at start of test (mo, mo, day, day, yr, yr, hr, hr, min, min, sec, sec).
Test Station Operator MOS/Skill Level	SYS	10	The test station MOS code with the skill identifier, if applicable.
Test Station Model Number	SYS	13	ATE model number.
Test Station Serial Number	SYS	4	ATE Serial Number.
Status	TPS	4	The status of each test step within the TPS, either GO or NOGO.
Test Step Lower Limit	TPS	12	The lower limit value of each test step within the TPS.

AMCOM DATABASE									
ID	TEST #	SERIAL #	LL	UL	MV	DIMS	STATUS	PART #	CALLOUT
2265	320000		0.001	1.8	4.5443E+37	DEGC	NOGO	13080451-029	R/R: A33, A29/A30, A32, IR IMAGER
2266	290000		-0.1	0.1	0.0445	DEGREES	GO	13080451-029	
2267	291000		-5	5	-1.8058	MRAD	GO	13080451-029	
2268	292000		-5	5	-1.3737	MRAD	GO	13080451-029	
2269	293000		-4.05	4.05	0.2721	MRAD	GO	13080451-029	
2270	294000		-3.13	3.13	1.3315	MRAD	GO	13080451-029	
2271	300000		-3.3	-0.5	-2.99	VDC	GO	13080451-029	
2272	301000		-2.49	2.5	-2.064	VDC	GO	13080451-029	
2273	302000		0.3	0.5	0.459	VDC	GO	13080451-029	
2274	303000		1	80	65.1008	PERCENT	GO	13080451-029	
2275	310000		0.27	1.1	0.4474	NONE	GO	13080451-029	
2276	310000		0.11	1.1	0.3632	NONE	GO	13080451-029	
2277	310000		0.02	1.1	0.2328	NONE	GO	13080451-029	
2278	310000		0.01	1.1	0.1868	NONE	GO	13080451-029	
2279	320000		0.001	0.88	17.5782	DEGC	NOGO	13080451-029	R/R: A33, A29/A30, A32, IR IMAGER
2280	320000		0.001	1.35	21.6957	DEGC	NOGO	13080451-029	R/R: A33, A29/A30, A32, IR IMAGER
2281	320000		0.001	1.8	25.6039	DEGC	NOGO	13080451-029	R/R: A33, A29/A30, A32, IR IMAGER
2282	150000		4854	5366	5100.6151	OHMS	GO	13080451-029	
Test Step Upper Limit			TPS	12	The upper limit value of each test step within the TPS.				
Test Step Measured Value			TPS	12	The measured value of each test step within the TPS.				
Dimension			TPS	4	Type of measurement made of each test step within the TPS (e.g., OHMS, VDC, etc.)				

Table 16: Structure of the SHELFP and AMCOM data warehouse.

The Relational Database System in SHELP

The database tables can be updated to include new platforms, systems and subsystems via the software interface. Tables will be dynamically created to allow the insertion of these new platforms. The database content will be ever changing, and the S-HELP software was programmed to accommodate that. Currently, the query options allow only a few fixed types of analysis due to the fixed database structure. In future endeavors, SHELP will be implemented with an adaptive query on a dynamic relational database management system, hence integrating more user defined analysis types for data association.

The relational model follows the system hierarchy of the Army Weapon Systems. Starting from a Platform (Apache) as shown in Figure 95 which is in the topmost table, there is a one-to-many relationship between the platform and the Systems within itself. In the Systems table, the subsystems are listed in data columns, with the numeric and noun representations sitting side-by-side in adjacent columns. These subsystems can be found as entry items in the maintenance test log data warehouse for the Apache platform.

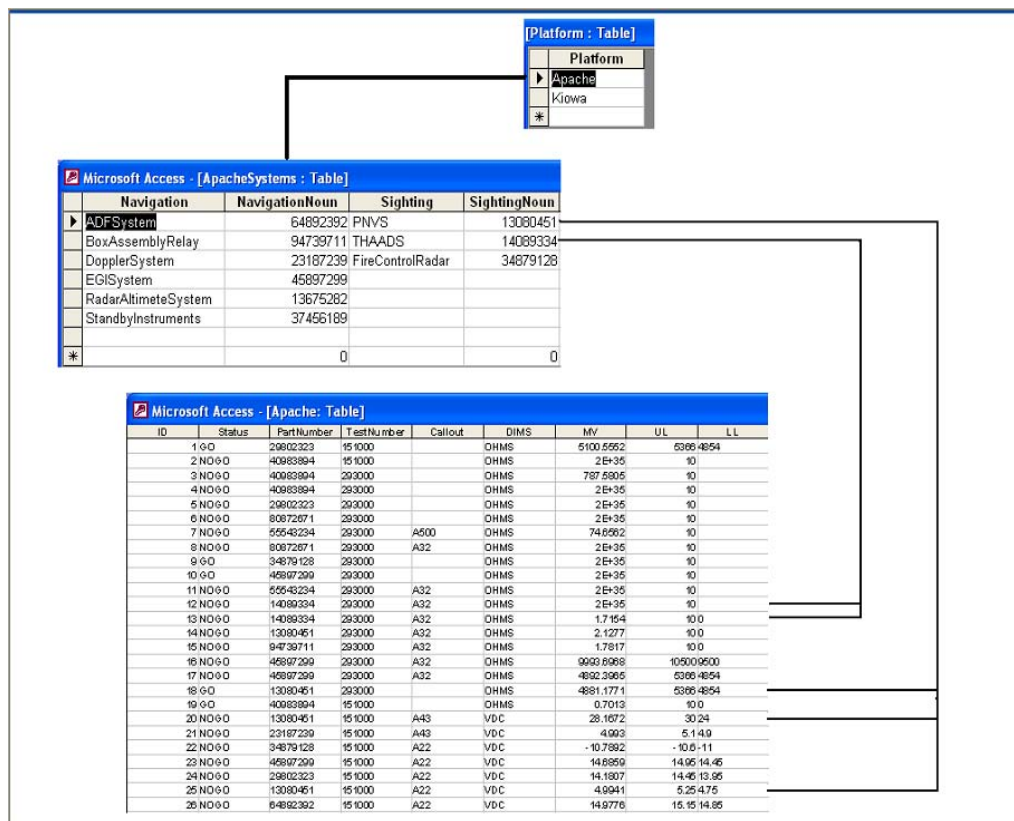
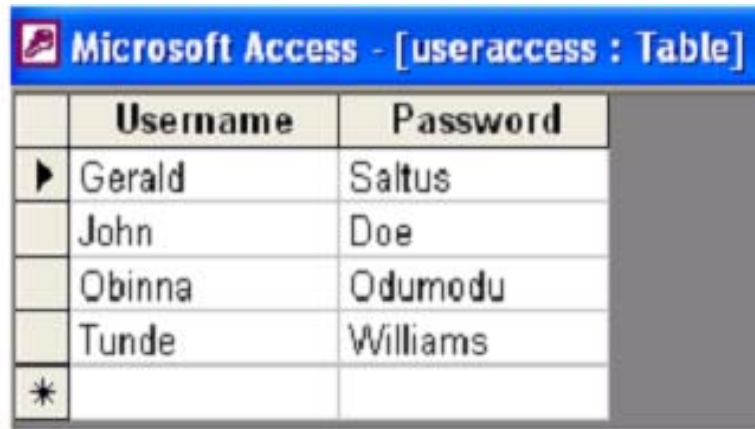


Figure 95: The relational model of the SHELP database

SHELP User Verification table

One more feature added in the database is the User account table. This implements a simple user verification concept, whereby Usernames and passwords are matched and verified during each login attempt. This is also an updateable database, where users can be added and removed, and Usernames and passwords can be modified.



	Username	Password
►	Gerald	Saltus
	John	Doe
	Obinna	Odumodu
	Tunde	Williams
*		

Figure 96: User verification table

User Roles

There are two existing users of the SHELP tool, the Regular User and the Administrator. The User model is an additional feature of the SHELP tool that will be greatly enhanced upon actual development of the final product using MSSQL server, which will utilize one of the many security models available. Since this is a proof of concept, the user model was not implemented with heavy security considerations. Hence, a simple user authentication process is performed in order to differentiate the regular user from an administrative user.

The regular user has data query privileges, so does the Administrator who has additional rights to modify different items within the SHELP database.

Query Structure and Bayesian modeling

The query structure of the SHELP tool is a downward data filtration process implemented in a tree structure within the source code implementation of SHELP. Once the data has been filtered down to the last two connecting nodes, winmine is used to analyze the data remaining in the one-to-one relationship of the last two nodes. This process is a mimic of the Bayesian models generated by winmine as shown in fig. The winmine toolkit was not used in the data filtration process because winmine is a stand alone software with no reference library provided to navigate through the nodes programmatically. Figure 97 shows the query tree of the Apache platform and the different query levels. Figure 98

shows the path followed for a query of THAADs subsystem's pass/fail history for test# 290000 in Iraq. Figure 99 shows the path followed for a query of the THAADs subsystem's pass/fail history in Iraq, for test# 290000. Although the query response will display the same values in both cases, on the display window, in the first case (fig.), SHELP will also produce information regarding other test numbers in Iraq and in the second case (fig.), SHELP will display information for other locations with test number 290000's result for the fore stated system.

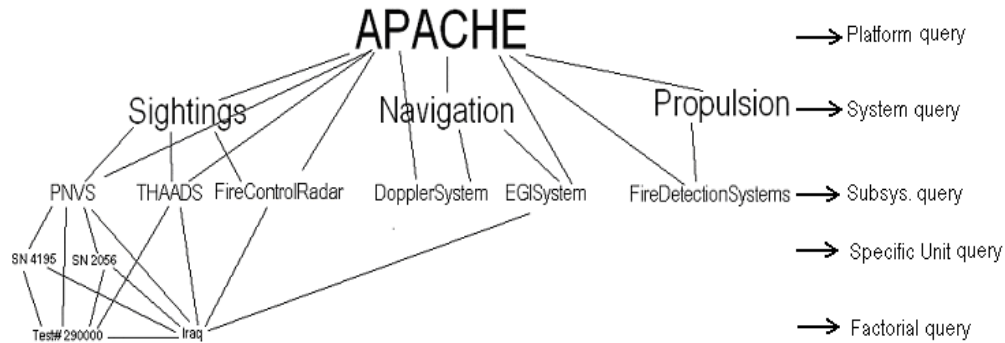


Figure 97

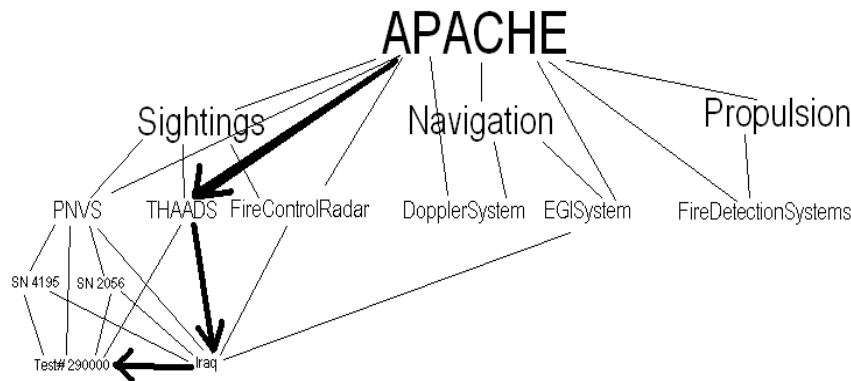


Figure 98

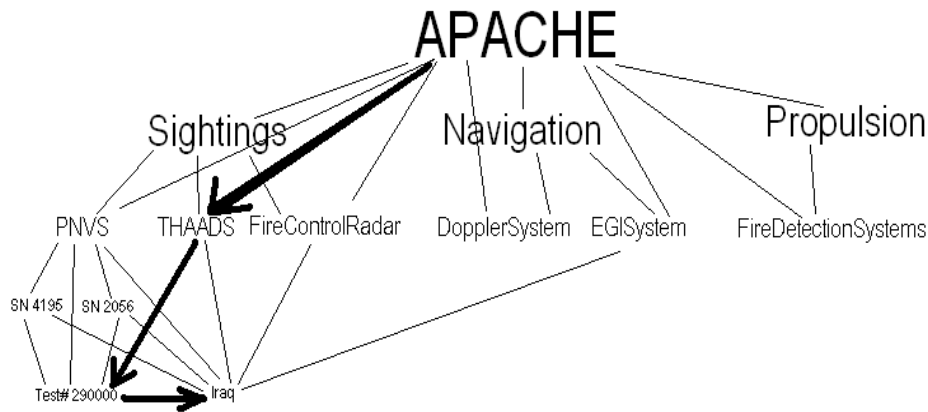


Figure 99

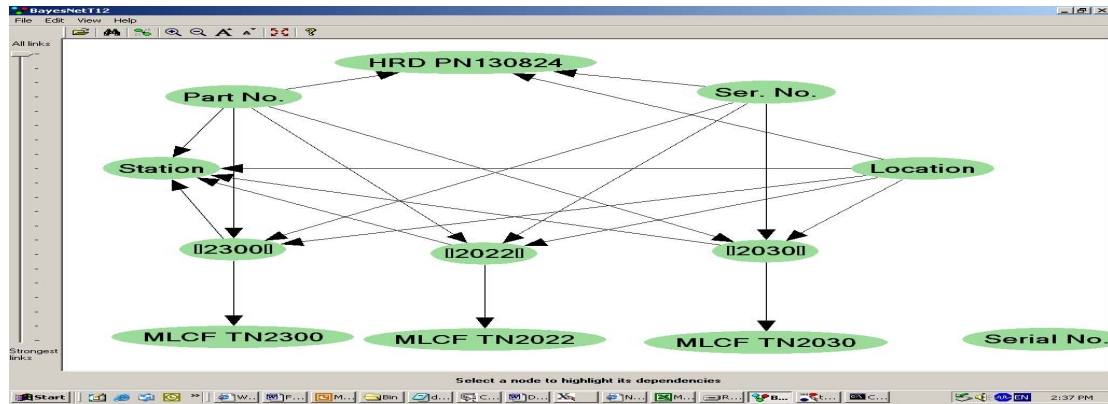


Figure 100: The S-Help query in winmine

System Architecture

The user interface was made to guide the user through the process painlessly. For use of this tool, there would be no training involved. We assume the target user is conversant with the platforms included in S-HELP, if not there are different levels of analysis available to meet the user's level of expertise. An analysis can be made from the Platform, System, and Subsystem level. If at all any aspect of the tool may become unclear, a user's manual is provided in the form of a help file for assistance.

The SHELP system data flow and user interface is illustrated in Figure 4. Once the software is initiated, a Login screen (Figure 101) will appear asking the user to specify his/her role.



Figure 101: S-Help login page

There are two options, (1) User (2) DB administrator. The user role does not require a username and password. However, the Administrator role requires a username and password. The Regular user will be directed straight to the query page. While the System Administrator will be directed to a page where he would be given the option to query the database or update the database. If he chooses the query option, he will be directed to the query page where he has a choice to make an analysis from any given level of a platform.

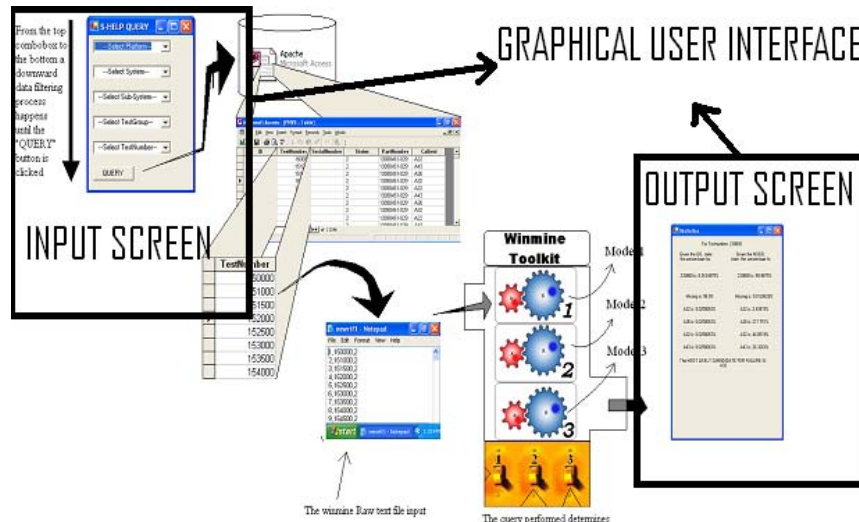


Figure 102: S-Help GUI Component

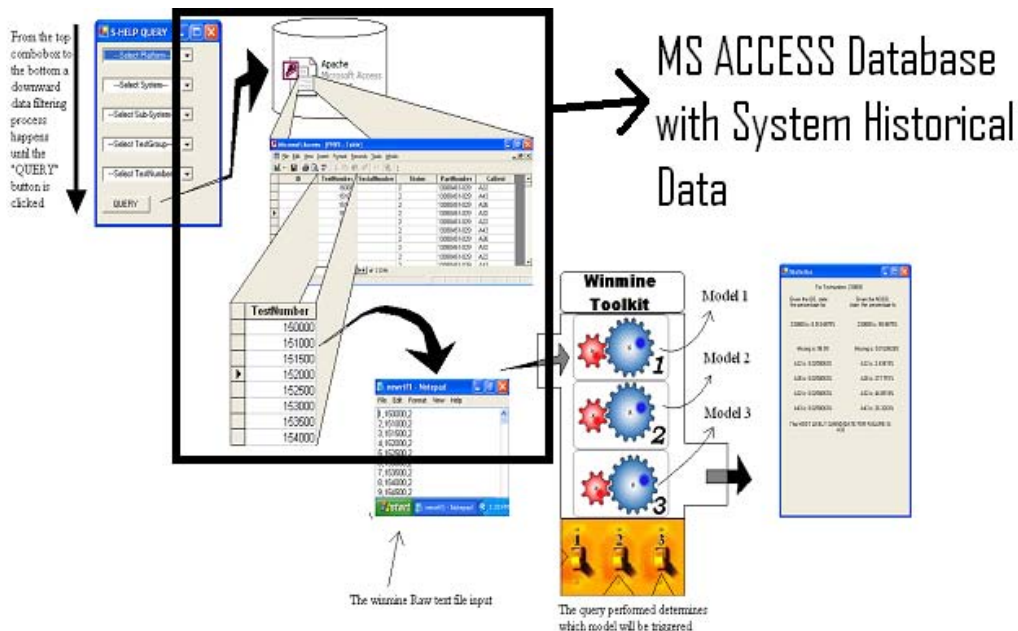


Figure 103: S-Help Database Component

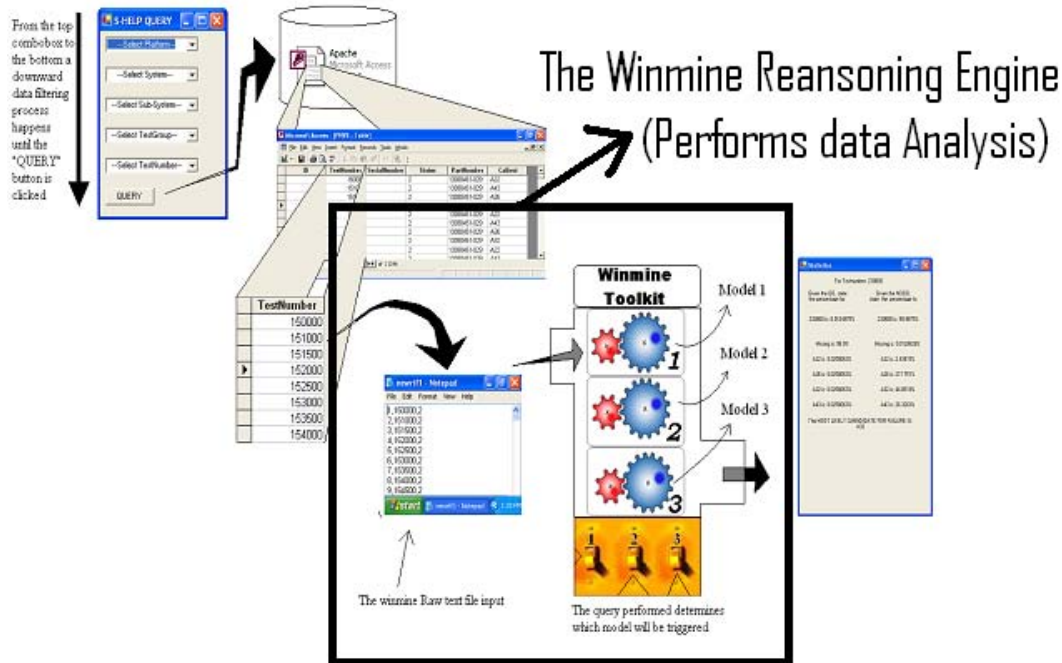


Figure 104: The S-Help Data-Mining Component

The result will be a percentage probabilistic report on the items associated with the user's query.

On this form, as illustrated in Figure 105 there are five combo boxes representing the depth of analysis starting from the overall platform, down to a system, then a subsystem, and if chosen, to a specific serial number. The test number, location and most likely candidate selections are additional query options.

The first combo box requires the user to select the Platform, e.g. Apache, Kiowa, Avenger. From the platform level, the user can query to see which subsystem fails/passes most frequently within the underlying platform.

The second box asks for the System within the Platform, e.g. Sighting, Propulsion, and Navigation. From this level the user can query the database to find out which subsystem is the most likely candidate for failure or success within the underlying system.

The third combo box asks for the subsystem within the system, e.g. PNVS, THAADS, or Fire Control. At this level an analysis can be made to find out the subsystem that fails or passes most often.

The fourth combo box is an optional combo box, which specifies unique serial numbers for the selected subsystem in the previous combo box. A query at this level will produce the serial number that fails/passes the most within the underlying subsystem.

The final two combo boxes are not system queries but factorial queries on the above system or part selections. Test numbers are to be selected in order to perform an analysis on the most likely test for failure/success, while locations are to be selected to perform an analysis on the most likely location of failure for the underlying system or part.

To pin point the LRU (Line Replaceable Unit) that might be the cause of a failure two check box selections are included at the bottom of the window's form labeled "Most Likely candidate for failure(LRU)" and "Recommended Solution."

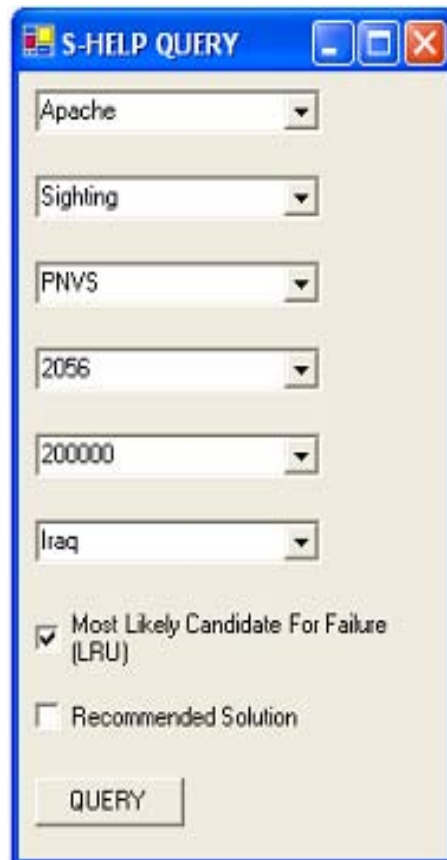


Figure 105: The Query Form

The Output Form

This software collects the probabilistic outputs of the queried data from the XMOD file, which is the output file from the WINMINE toolkit. These outputs are then text formatted for the user's understanding and printed to the output form as illustrated in Figure 106.

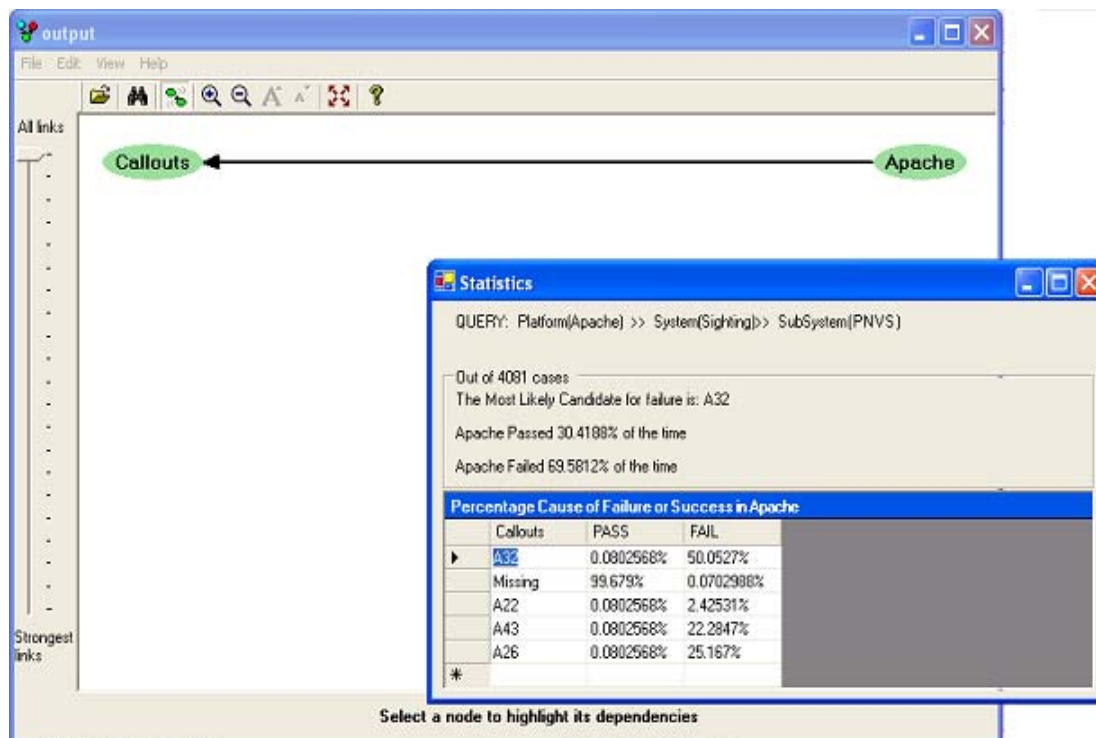


Figure 106: The Output Form

Source Code Implementation:

The S-Help tool is a Microsoft Visual Basic .Net windows and web application project. In Visual basic, windows form classes, components, and event handlers are pre-coded. So the working functions of the SHELP software lie within a global module file included in the S-Help project. Altogether, there are 26 working functions in the code that perform tasks such as record, fields and item retrieval, text file creation, reading and writing, XML file creation, navigation, data retrieval and parsing. In addition, Macro and shell execution was necessary to communicate with the stand-alone WINMINE toolkit. There were different algorithms used to parse and filter text and XML files, such as sorting, grouping, searching and recursion. Concepts such as overloading, optional arguments, inheritance, dynamic creation for display controls, and referencing was also necessary to carry out the tasks required of the S-Help tool.

Conclusions

SHELP currently uses non-enterprise third party software, such as MSAccess and Winmine Toolkit, which pose different limitation problems. Plus, the SHELP web application uses Internet Information Services 5.1 which is not user extensive.

Current technology limitations

MSAccess: (1) A table within excel will allow only 65,535 records. (2) A query cannot be performed through a relationship model created in MSaccess. (3) In order to analytically process data stored in MSAccess, an add-in software is required. (4)

MSAccess cannot be implemented as a Dynamic Relational Database Management System, because it depends on the SQL query syntax, which is mainly a static query representation and still the problem exists of MSAccess possessing no analytical tool.

Winmine Toolkit: (1) This is a stand-alone tool that does not ship with a reference library. So, in order to programmatically execute tasks in winmine, macros and self-generated xml datafiles were generated which are not flexible solutions for an adaptive system.

Internet Information Services (IIS): (1) Has a ten user access limit.

Microsoft SQL Server will serve as the platform that resolves many of the setbacks fronted by the previous SHELFP technologies. It includes an analysis component, a programming layer, and provides data accessibility to limitless users.

Microsoft SQL Server's data mining technology, Analysis services, helps users analyze data in relational databases and multidimensional OLAP cubes to uncover patterns and trends that can be used to make predictions and accept and adapt to new incoming data. The data mining capabilities in MSSQL Server are integrated tightly with both *relational and *OLAP data sources.

Microsoft SQL server is also accessible through web interfaces such as ASP.NET. Microsoft SQL Server in its flexibility will allow new factors such as costs, price, time, seasons, etc. The consideration will be endless, and other database field can be included in the OLAP cube on demand.

Student Project 2 – Intelligent Sensors

This section details work on an embedded hardware project to implement a material fatigue counter, which was investigated by students under this project. Students involved with this project include:

Ms. Rickenya Goodson, EE Undergraduate
Mr. Kristopher Cross, EE Undergraduate
Ms. Tiffany Tarver, EE Undergraduate
Mr. Benjamin Douglas, EE Undergraduate

Background

It is a stated objective of the funding agency to investigate and adopt practices of maintaining equipment and weapon platforms based on the current physical state or condition of the systems, as opposed to following traditional regular maintenance procedures and schedules. This change, which shifts the emphasis from preventative and reactive methodologies to a more proactive based philosophy, is commonly known as Condition Based Maintenance (CBM).

With this philosophy in mind, this project examines techniques for non-invasive assessment of the physical condition of system components. Techniques for sensing component fatigue and vibration modes are discussed. A prototype system to implement and test these methodologies is examined. The system is based on the integration of small accelerometer sensors, field programmable gate arrays (FPGAs) and wireless radio frequency identification (RFID) technologies.

“Fatigue” is a failure mode characterized by slow crack growth over a long period of time. The system stress initiating the crack, and causing crack propagation, is typically much less than the yield stress of the engineering material. S/N (Stress vs. Number of Cycles) curves are utilized by the design engineer to keep the component stress below the fatigue stress limit. But, material deterioration due to diffusion, corrosion, and creep eventually allows cracking to initiate. In static loading situations, the effects of fatigue are limited, but for components that exist in dynamic and rotating machinery the effects are dominant. Continuous vibrations excite system natural frequencies, and dramatically accelerate the process of fatigue.

A direct count of the fatigue cycles (vibrations) experienced by a component coupled with historical statistics can provide a good indication of remaining useful life. This is evidenced by the ubiquitous use of S/N curves in mechanical design. In addition, analysis and tracking of characteristic natural frequencies can provide an indication of crack initiation and growth in system components, (Askeland, 1989).

Embedded Fatigue Sensor

A conceptual electronic monitoring system based on the integration of small accelerometer sensors, field programmable gate arrays (FPGAs) and wireless radio frequency identification (RFID) technologies is considered and illustrated in **Figure 107**. Component characteristic frequencies due to the vibration modes of the UUT, are measured by the accelerometer sensor. Algorithms utilizing signal processing algorithms, including the Fast Fourier Transform (FFT), and vector correlation can be utilized for analyzing the sensor output. This could help diagnose and prognosticate conditions potentially leading to decreased component performance.

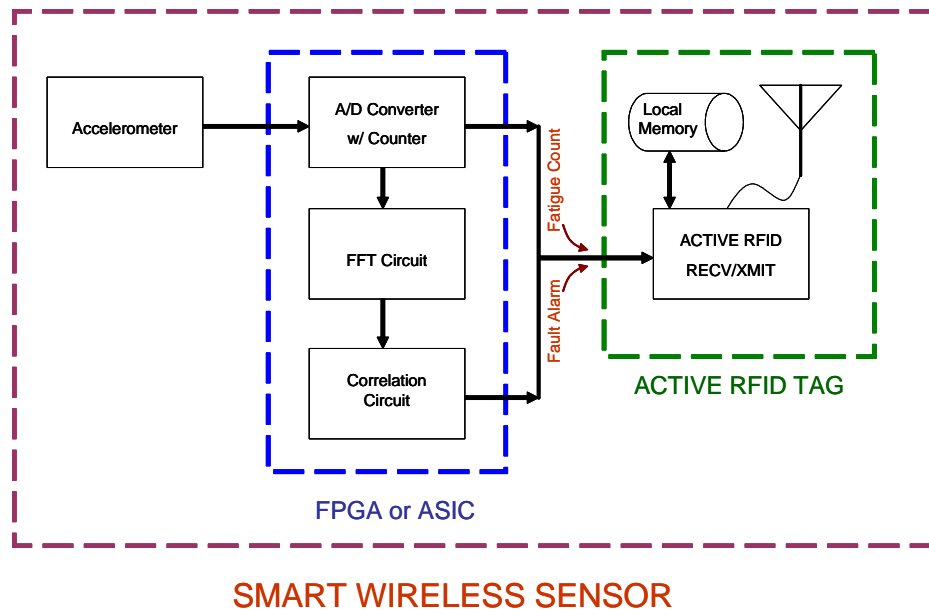


Figure 107: Concept for Prototype Sensor

The accelerometer can be one of a number of available COTS piezoelectric acceleration sensors with a range to at least 20 kHz. The signal processing algorithms can be implemented utilizing a low-cost Field Programmable Gate Array (FPGA) technology, (see www.xilinx.com, and www.altera.com for details) which allows custom combinational and sequential logic designs to be readily programmed into hardware. Once a final design has been developed the signal processing portion could be implemented into a custom ASIC microchip for a very small footprint. An active RFID tag can be utilized for the prototype system. These tags allow access to their onboard memory for I/O flags, and data logging from external circuitry. The tags are capable of being remotely queried (scanned) similar to a passive tag, or can be programmed to wake up at regular intervals and broadcast select memory locations to a remote (< 400 ft) receiving station. Other leading edge wireless options include the “Memory Spot” by Hewlett Packard (www.hp.com), and the “Enterprise Dot” by AXCESS, Inc.

(www.axcessinc.com), both of which offer embedded processing capability with a wirelessly enabled memory system in a very small package.

The prototype “smart” wireless sensor system consists of a number of subsystems as illustrated in Figure 108. The flow of data from the accelerometer or strain gage sensor is shown. The analog output signal from the accelerometer is passed to an A/D converter, possibly through an amplifier and/or filter circuit. The A/D converter digitizes the signal, and counts cycles for use in tracking the fatigue count. The fatigue count is sent to the RFID memory for addition to previous counts, and to enable wireless access to cumulative count totals. Also, the digitized signal is sent to the FFT calculation circuit, via a memory buffer that feeds the FFT circuit when the buffer is full. The signal vectors in frequency domain format that are generated by the FFT circuit are fed into a correlation circuit. There they are compared to benchmark “NORMAL” operation vectors. Correlation values below a user defined threshold (e.g. 50%) set an alarm signal that is forwarded to the RFID circuit. The alarm “wakes up” the RFID circuit, causing it to transmit, and notify the receiver of an IMMINENT FAULT, or a perceived FAULT.

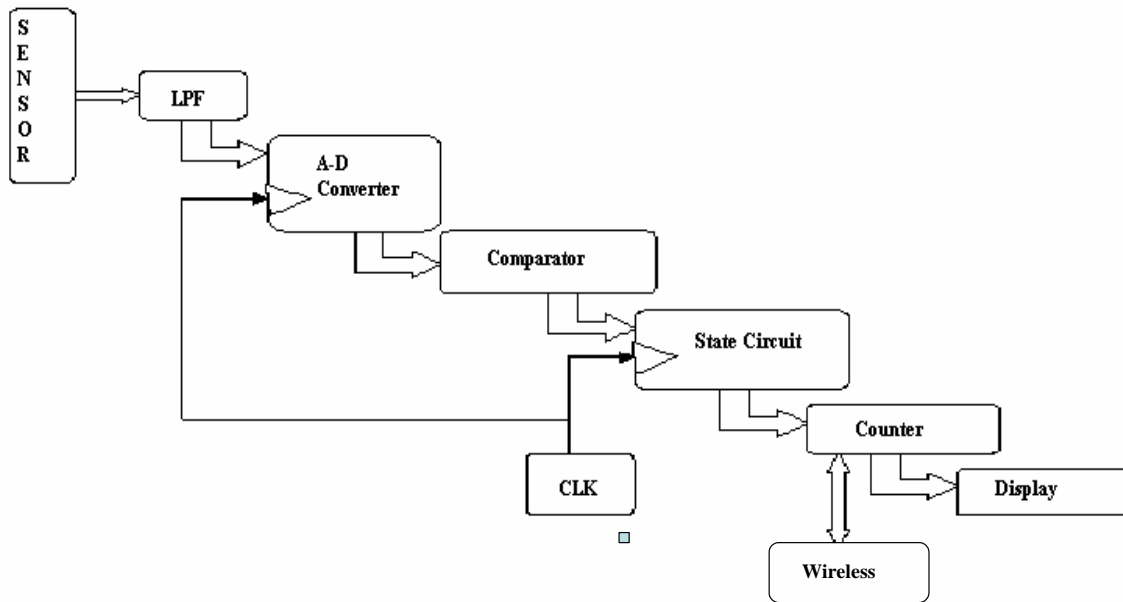


Figure 108: Sensor Data Flow Diagram

Sensor Selection and Placement

There are two main types of devices to consider for sensing of the vibration characteristics of system components, accelerometers and strain gages. Both sensors

come in a variety of size and package configurations, some of which are small enough to be easily placed on components as small as a circuit card.

Accelerometer sensors work by providing an output voltage proportional to the detected acceleration of the sensor. They offer many benefits; low power consumption, small footprints for piezoelectric models (on the order of 2 mm²), multi-axial sensing (1 to 3 dimensions) coupled with the capability to detect a wide range of accelerations (e.g. micro-g's through 100s of g's), and many offer dynamic shock protection up to 1000s of g's.

Strain gages sensors work by sensing the change in resistance of a serpentine arrangement of wires directly attached to a component. They are hooked through a Wheatstone bridge, and provide a direct measurement of the amount of strain present in the component, which is directly related to the stress through Hooke's Law relationships. The benefit is that very accurate measurements of actual stress can be drawn from them. However, there are at least two main drawbacks that make them unsuitable for use in this situation; 1) relatively high power consumption (compared to accelerometer), and 2) the placement of the gage is critical. If the gage is improperly placed in a region of low stress, the character of the dynamic stress is lost.

In some situations, particularly on a circuit card, there may not be room to directly attach a strain gage in the high stress region, which typically occurs near the bus connection. Moving the gage to a region of the card with more available space may cause the sensor to miss localized stress tensors. Similarly, it is desirable to place accelerometers in areas on the card that undergo the most movement, but as can be seen from the modal simulations, there are a number of candidate locations. For these reasons, the accelerometer is the sensor of choice, and its placement is not as critical as that of the strain gage.

For the purposes of this work, a board (Freescale MMA7261Q) mounted accelerometer, with an easily configurable interface was utilized for testing as shown in **Figure 109**. The accelerometer itself operates on two axes, and can detect in the range of -18g to +18g.



The diagram illustrates the internal architecture of the AD673. It features an 8-bit current output DAC and an 8-bit SAR (Successive Approximation Register). The SAR is controlled by an internal clock (INT CLOCK) and a data enable signal. The DAC output is connected to the SAR. The SAR output is an 8-bit digital word, with the most significant bit (MSB) and least significant bit (LSB) indicated. The output is also connected to a data enable signal. The diagram also shows the analog input section, which includes a 5k resistor and a bipolar offset control signal. The digital common and convert signals are also shown.

Figure 110: 8 bit A/D Converter

The acceleration signal is sampled well above the indicated Nyquist frequency to ensure accurate replication of the component vibration. This is illustrated in Figure 111. Furthermore, the figure illustrates the threshold cutoff line for counting fatigue cycles on the component. This voltage level will be used to trigger the fatigue counter circuitry.

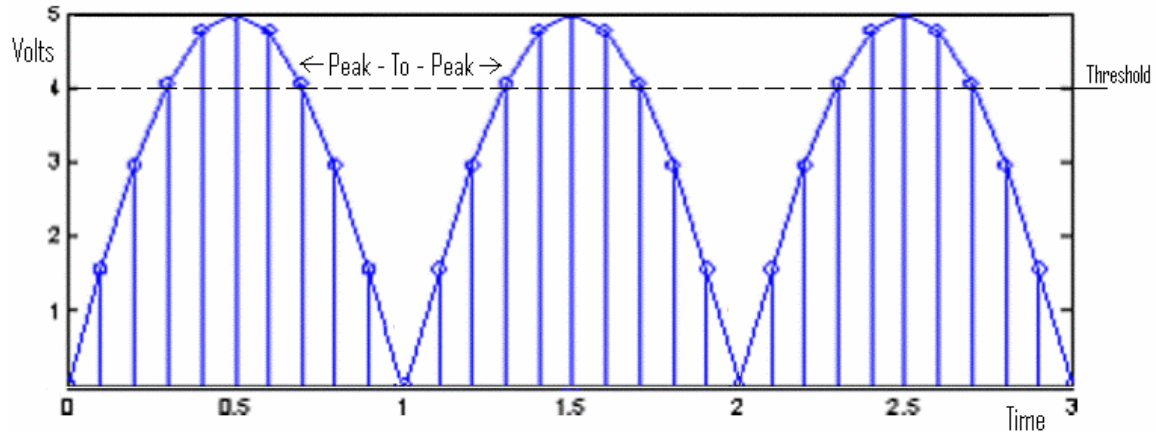


Figure 111: Representative Sensor Output

Fatigue Counter Development

An eight bit comparator circuit was developed to implement the threshold trigger described above. A pre-set threshold value was directly encoded into the circuit. The output from the comparator (e.g. $<$, $>$, $=$) was used to determine the state of the counting circuitry. The state diagram depicted in Figure 112 illustrates the transition between states of the counter. The signal will be in one of two positions, above the threshold, or below. As the instant the signal first breaches the threshold reference level, the state circuit transitions from S0 to S1. As long as the signal stays above the threshold, the state doesn't change. Only when the signal drops below the threshold does the state transition back to S0. The S1 to S0 transition triggers the fatigue counter to increment.

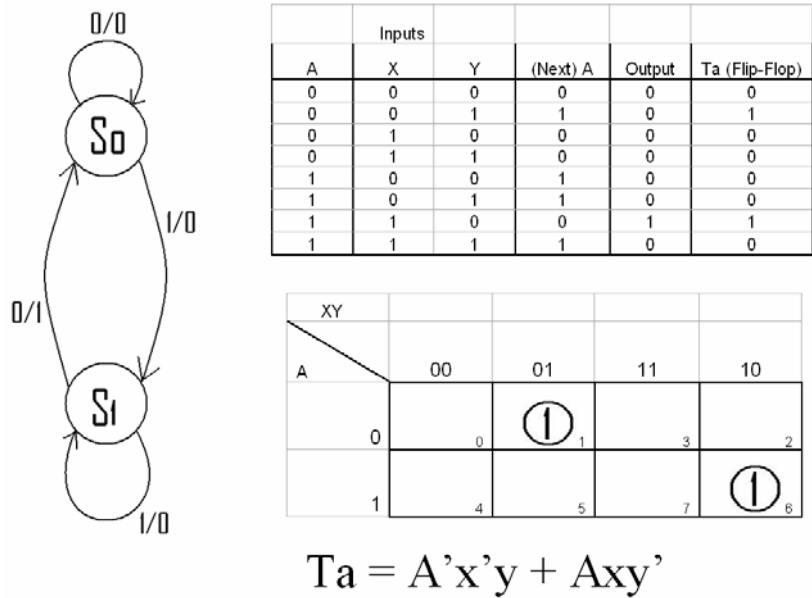


Figure 112: State Circuit for Fatigue Counter

The state circuit was implemented with combinational logic, and a T type flip flop. Figure 113 illustrates a pictorial schematic of the state circuit with validation by simulation.

A counting circuit to track and report the number of active fatigue cycles was created by cascading a number of decade counters. This is pictorially represented in Figure 114, which also schematically illustrates the LCD displays that are used for debugging purposes, and defines the function of the circuit.

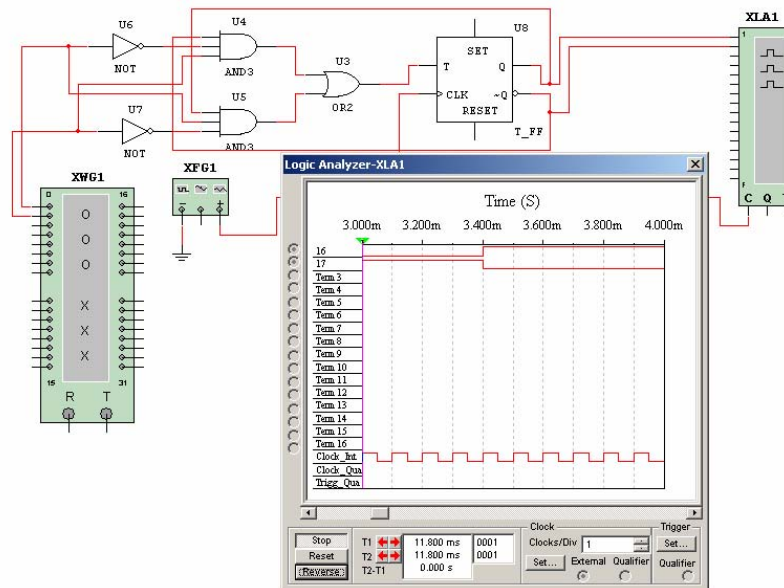


Figure 113: State Circuit Validation

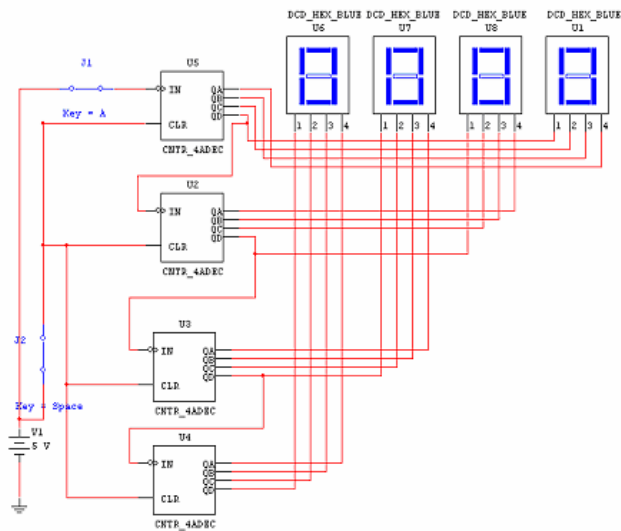


Figure 114: Counter and Display Circuit

With reference to Figure 108, all the required circuitry below the A/D converter in the signal flow path was implemented on an FPGA platform as shown in Figure 115. The output from the A/D converter feeds into the comparator circuit, which in turn drives the state circuit as described above, and provides triggers to implement the cascading counter circuitry. The output (RHS) goes to an LCD display. In the field, though, this output will be stored to non-volatile memory, and utilized to track the fatigue cycles across power cycles.

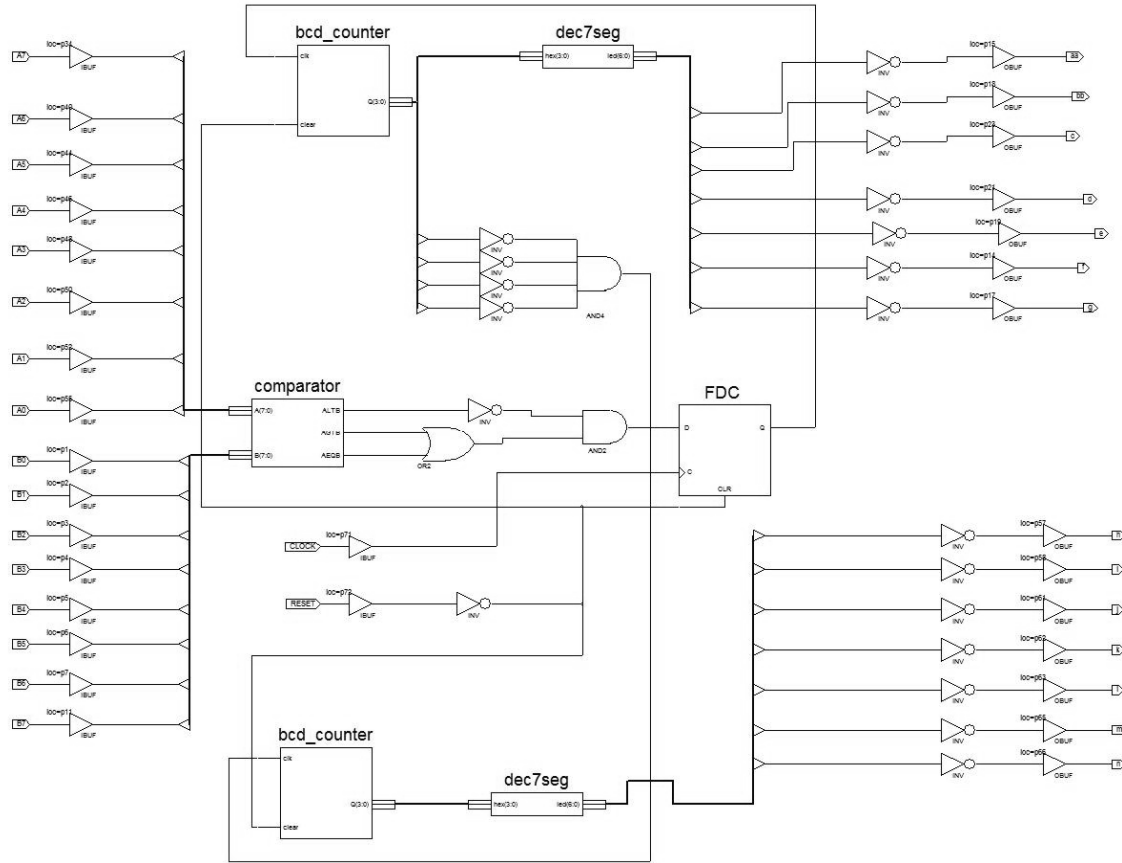


Figure 115: Composite Digital Circuit

Figure 116 shows the FPGA prototyping board that was used to implement the composite digital circuit. The FPGA was programmed via the available Xilinx ISE software. Use of the prototyping board enables ease of interface to the peripheral devices and ancillary circuitry.



Figure 116: FPGA Prototype Board

Active RFID

The prototype circuitry that was developed and implemented on the FPGA, was interfaced with an active RFID (Radio Frequency Identification) tag. These tags allow access to their onboard memory for I/O flags, and data logging from external circuitry. The tags are capable of being remotely queried (scanned) similar to a passive tag, or can be programmed to wake up at regular intervals and broadcast select memory locations to a remote (< 400 ft) receiving station. The tags and accompanying support equipment were acquired from AXCESS, Inc. (www.axcessinc.com). The Active tag supports two modes: 1) it can be programmed to wake, and send at regular intervals, or 2) it can be queried by an activation device at any time. In addition, the tag has the capability of storing and transmitting data in its non-volatile memory system. It is desired to store the current fatigue count in this memory system for access externally. The equipment consists of an Activation device, which in essence “queries” the active ID tag through a field generating antenna. The activation device can interface through a remote PC via a standard serial connection to allow for software initiated queries on the tag. The RFID tag, activation device, and field generating antenna are pictured in Figure 117. The devices meet the following specifications:

- Active RFID
- On board memory
- Small dimensions of 3”x2”x1”
- baud rate of 19.2 kbps
- Active RFID Field Generating Antenna
- Activation Range of 35 to 40 ft.
- Light weight and mobile

- Activator for the Generating Antenna
- Connects to a PC through a DB9 serial cable.



Figure 117: RFID Tag and Activation Equipment

When the RFID tag broadcasts in either mode, its signal is picked up by the RFID Receiver, which is shown in Figure 118. The RFID receiver meets the following specifications, and also links to a remote PC via a serial (DB9) connection, or through an Ethernet protocol network via RJ45 interface. This enables software monitoring of the RFID tag.

- ActiveTag Network Receiver
- Transmission Range: Up to 35 feet
- Dimensions: 1"x4.5"x5.5" (Receiver) 8"(Antenna)
- Operating Temperature: -40°C to +85°C



Figure 118: RFID Receiver

Intelligent Sensor Results

A prototype intelligent sensor to enable fatigue counting was created. It was based on the above specification, and assembled and tested undergoing a number of iterations. A team of AAMU students helped to construct the prototype for their Senior Design Project. The team consisted of the following students: Ms. Rickenya Goodson, Mr. Kristopher Cross, Ms. Tiffany Tarver, and Mr. Benjamin Douglas. Figure 119 illustrates the conglomerate prototype, being demonstrated by the AAMU students.

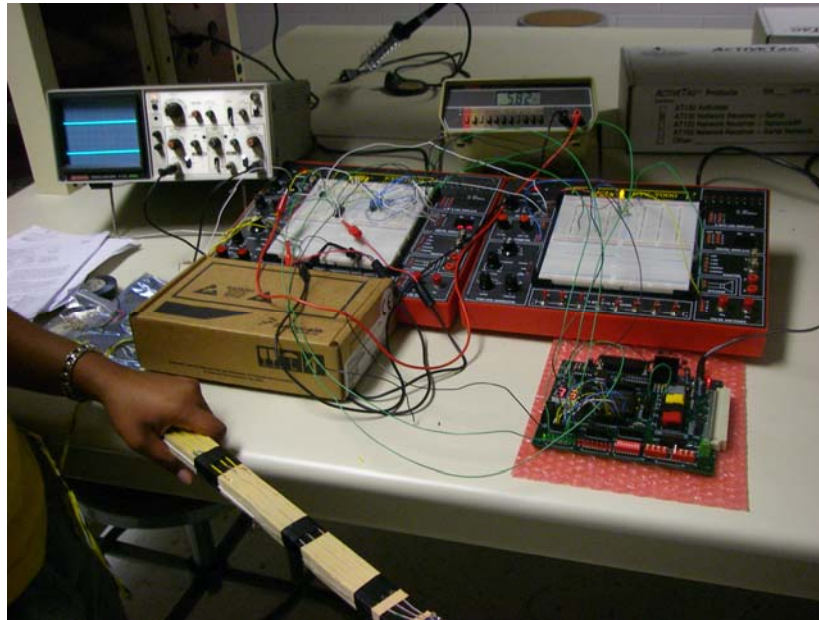


Figure 119: Student Demonstration of Intelligent Sensor

Intelligent Sensor Conclusions

A prototype intelligent sensor to enable fatigue counting was created. It consists of the following sub-systems:

- a sensing component implemented with an accelerometer;
- an A/D converter, which was interfaced with the accelerometer to enable digital manipulation of the sensed vibration cycles;
- a comparator circuit, which compared the level of the incoming signal to a pre-specified threshold value; a state circuit, which was triggered by the comparator circuit, enabling the tracking of only full cycles of the vibration signal;
- a counting circuit that incrementally increased the previously stored fatigue cycle count for each full vibration cycle encountered by the sensor circuit;
- and an active RFID circuit that allowed for non-volatile storage of the current fatigue cycle count, and enabled remote query and/or regular broadcasts of the data to a remote receiver, which was coupled to a PC, thus enabling software tracking of the wear on the component being monitored.

The results obtained from this work is encouraging, and indicate that using fatigue analysis to track component wear by utilizing frequency domain analysis of component vibration profiles may be very useful in characterizing overall system health.

Pattern Recognition Course

As part of the project, a course in Pattern Recognition was developed within the Department of Electrical Engineering. The course is designed for advanced senior students, and graduate students. A survey of similar courses offered at a number of engineering schools throughout the country was conducted. This helped identify the most popular topics and texts that are commonly in use. As yet, the course does not have permanent classification since it has not been screened by the University Standards Committee. It is presently classified as a “Special Topics” course, EE 490 for undergraduate students, and EE 590 for graduate students. The School of Engineering is beginning a new Master’s level program (Fall 2008) in “Materiel Engineering,” offering core concentrations in Electrical, Mechanical and Civil Engineering. The degrees will be given in Master’s of Engineering. It is anticipated that the pattern recognition course will play a strong role in the EE core component for educating the graduate students. The Department of Electrical Engineering participates as a core portion of the program. A detailed outline of the Pattern Recognition course is presented below, followed by a description of the graduate program.

Course Description

Alabama A&M University

Department:	Department of Electrical Engineering
Course Number and Title:	EE 490/590: Pattern Recognition
Designation:	Senior Elective Course
Catalog Description:	EE 490/590 – Pattern Recognition, 3 hrs. This course covers both basic and advanced techniques in Pattern Recognition. The goal of the course is for students to gain theoretical and practical understanding of various classification techniques and their application to real-world problems.
Prerequisites:	Familiarity with basic probability and linear algebra, (MTH 280 and MTH 453 or equivalent), or permission of Instructor. (Offered As Needed)
Required Texts:	1. Duda, Hart and Stork. <i>Pattern Classification</i> . John Wiley and Sons, Inc. New York, 2001. ISBN: 0-471-05669-3. and 2. Stork and Tov, <i>Computer Manual in MATLAB to accompany Pattern Classification</i> . John Wiley and Sons, Inc., New York, 2004. ISBN: 0-471-42977-5.
Reference:	C.M. Bishop. <i>Neural Networks for Pattern Recognition</i> . Oxford University Press. 1995. Judea Pearl. <i>Causality – Models, Reasoning and Inference</i> . Cambridge University Press, 2000.
Instructor:	Dr. Kaveh Heidary, Professor of Electrical Engineering
Office and Office Hours:	Room 209 ETB; MWF 9:00 a.m. to 1:00 p.m.
Contact:	(256) 372 – 5587; kaveh.heidary@aamu.edu
Course Learning Outcomes:	Upon completion of the course, students will be able to

Course Learning Outcome	Program Outcome	Assessment Tool
1. Understand the concept of feature detection and classification	a	Homework, Quiz, Exams
2. Understand conditional probability and Baye's rule	a, b, e	Homework, Quiz, Exams
3. Understand random vectors, correlation, covariance	a, b, e	Homework, Quiz, Exams
4. Exposure to decision trees and neural networks	a, b, e, j, k	Projects,
5. Exposure to topics in unsupervised learning and clustering.	j	Projects,
6. Develop MATLAB applications.	a, e, k	Computer Projects

Topics Covered:

1. Introduction
2. Statistical Decision Theory
3. Parameter Estimation
4. Kth Nearest Neighbor
5. Linear Discriminant Functions
6. Support Vector Machines
7. Neural Networks
8. Decision Trees
9. Stochastic Methods
10. Feature Selection
11. MATLAB software.
12. Applications
13. Programming Projects

Class Schedule: Monday and Wednesday 4:00 to 5:20 p.m.; Room 223 ETB

Contribution of course to Criterion 5:

There are five main objectives in this course: 1) Provide students with an overview of the concepts of feature detection and classification, (2) Reinforce the concepts of conditional probability, and Bayes rule. (3) Introduce the topics of random vectors, correlation and

covariance, (4) expose students to real-world applications that require decision trees and neural network models, (5) expose students to advanced topics in pattern recognition, e.g. unsupervised learning and clustering, (6) develop and implement algorithms in MATLAB programming language. The course serves as an advanced senior level, and introductory graduate elective course in electrical and computer engineering science.

Relation of course to Program Outcomes:

(a) an ability to apply knowledge of mathematics, science, and engineering	X
(b) an ability to design and conduct experiments, as well as to analyze and interpret data	X
(c) an ability to design a system, component, or process to meet desired needs	
(d) an ability to function on multi-disciplinary teams	
(e) an ability to identify, formulate, and solve engineering problems	X
(f) an understanding of professional and ethical responsibility	
(g) an ability to communicate effectively	
(h) the broad education necessary to understand the impact of engineering solutions in a global and societal context	
(i) a recognition of the need for, and an ability to engage in life-long learning	
(j) a knowledge of contemporary issues	X
(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.	X

Computer Usage:

Students will develop and utilize a number of MATLAB programming projects. These will be coupled with existing MATLAB library functions as provided in the accompanying text and course materials. Students will need an account to use the EE lab computers. Computer based projects will be assigned and constitute a part of the grading policy.

ABET category content as estimated by faculty member who prepared this course description:

Engineering Science: 3.0 Credits 100%

Prepared by: Dr. K. Heidary

Date: February 27, 2008

Grading Policy

- Quizzes:** Occasional quizzes will be given to ascertain students comprehension of class assignments and lecture materials.
- Projects:** Programming projects involving data parallelism and evaluation of elapsed time studies on multiple HPC networks and platforms will be conducted throughout the semester. Brief reports on the projects that include the computer codes, simulation schemes, results and discussion of the results are required to meet this segment of the grading policy.
- Periodic Examinations:** A total of two examinations, a Midterm and a Final, of equal weight will be conducted during the semester. Both exams will be fully considered. Questions given in these examinations will be based on the homework assignments, projects and the problems that are discussed in the classroom
- Grades**

1. Quizzes and Project Work	40 %	Grade A: 90 % and above
2. Midterm Exam	30%	Grade B: 80 to 89 %
3. Final Exam:	30 %	Grade C: 70 to 79 %
		Grade D: 60 to 69%
Total:	100 %	Grade F: Less than 60%

Graduate Program Description

MASTER OF ENGINEERING in MATERIEL ENGINEERING

DEPARTMENT OF CIVIL ENGINEERING

Pabitra K. Saha, Ph.D., P.E., Chair
Room 305 ETB

DEPARTMENT OF ELECTRICAL ENGINEERING

Kaveh Heidary, Ph.D., Chair
Room 214 ETB

DEPARTMENT OF MECHANICAL ENGINEERING

Mohamed A. Seif, Ph.D., P.E., Acting Chair
Room 307 ETB

The Departments of Civil, Electrical, and Mechanical Engineering collectively offer a graduate program leading to the Master of Engineering (M.Eng) degree in Materiel Engineering. Materiel is defined as the equipment, apparatus, and supplies used by an organization. Materiel engineering involves the design, production, test and evaluation, distribution, operation and support, and ultimate disposition of man-made equipment, apparatus, and supplies, and, as such, is highly interdisciplinary.

OBJECTIVE

The general objective of this program from AAMU is to provide a curriculum of the highest quality for post-baccalaureate learning opportunities in this highly important area. Specific objectives are as follows:

- Provide students with a solid foundation in Materiel Engineering, as well advanced studies in areas of Civil, Electrical, or Mechanical Engineering
- Provide a continuum of study leading to a graduate degree for qualified students at AAMU completing a baccalaureate in engineering
- Provide a program of study that also accommodates the needs and limitations of mature adults engaged in full-time employment
- Accommodate the admission of persons holding bachelor's degrees in non-engineering areas, using selected background courses
- Respond to the anticipated requirement of making the master's degree, or the equivalent, as an entry-level qualification for attaining professional licensure.

ADMISSION REQUIREMENTS

This program is intended for individuals holding a bachelor's degree from a regionally accredited institution in any area of engineering or a closely related discipline. To formally pursue the Master of Engineering degree, the student must apply to the Graduate School of AAMU for Regular Admission, meeting the requirements for either Unconditional or Conditional Status. To take individual courses without seeking the graduate degree, persons may be admitted in a Non-Degree Graduate Status.

Regular Graduate Admission Applicants for admission to the Graduate School of AAMU must provide transcripts from each post-secondary school attended, as well as a transcript of the Graduate Record Examination (GRE). They must also provide two letters of recommendation and submit details of any professional work experience. Students from non-English speaking countries are required to have a minimum score of 500 on the Test of English as a Foreign Language (TOEFL).

Unconditional Admission. To be admitted with Regular Status to the Master of Engineering program, an applicant must have

- Earned a bachelor's degree in an engineering program that had ABET accreditation at the time of graduation,
- Earned an overall Grade Point Average (GPA) of at least 3.00 on a scale of 4.00, or have passed the National Council of Examiners for Engineering and Surveying (NCEES) Fundamentals of Engineering Examination, and
- Minimum GRE scores of 600 on the quantitative portion and 1000 on the combined verbal and quantitative portions.

Conditional Admission. Applicants who do not meet one, but no more, of the three above-listed requirements for unconditional admission may be admitted on a conditional basis under the following conditions:

- GPA or GRE Deficiency. Persons with a bachelor's degree in engineering may receive Conditional Admission provided their GPA is at least 2.5 on all undergraduate engineering courses attempted. This condition also holds for individuals with a GRE deficiency, including those who have not taken the GRE.
- Degrees in Other Fields. Individuals with a bachelor's degree in physics, mathematics, computer science, chemistry, or other fields closely related to engineering may receive Conditional Admission provided they have completed the following:
 - Mathematics through Differential Equations
 - Series in General Chemistry
 - Series in Calculus-based Physics
 - Introductory Computer Programming
 - Other courses prescribed by the advising engineering department, including specific topics stated as prerequisites for intended graduate courses.

Persons not meeting these minimum preparatory studies are advised to register as undergraduate students while completing this background.

Under Conditional Admission, a student may complete up to 15 semester hours credits recommended by the department advisor. A student in Conditional Admission is required to earn a minimum of 'B' grade in these courses to progress to Regular Admission; otherwise, the student will be dismissed from the School of Graduate Studies.

Non-Degree Graduate Status Professionals in the community may desire to take certain courses for graduate credit without the full formalities of applying for the graduate program. The AAMU School of Graduate Studies provides for the admission of such persons in a non-degree graduate status. To qualify for this admission, individuals must be a graduate of a

regionally accredited institution in the United States with a 2.5 or higher undergraduate GPA. They must also have the prerequisites for any course pursued at AAMU.

To pursue graduate engineering courses using the non-degree graduate status, an application must first be submitted to the School of Engineering and Technology. This is to ensure that the applicant has the required background for the desired course(s). If approved, the application will then be passed on to the School of Graduate Studies for processing.

Up to nine semester hours of graduate credit may be earned while in the non-degree status. Later, if the individual applies for and receives regular admission to the program, applicable credits earned with a grade of 'B' or better may be applied to meeting the program requirements.

DEGREE COMPLETION REQUIREMENTS

Academic Credit: A minimum of 30 semester hours credit in graduate-level courses will be required. These are distributed as follows:

Core Courses	12 s.h.
Discipline Specialization Courses	9 s.h.
Approved Electives	6 s.h.
Master's Project	<u>3 s.h.</u>
TotalAcademicCredit	30 s.h

1. Core Courses:

GEN- 601 Life-Cycle Design Engineering	3 s.h.
GEN -602 Product Assurance Engineering	3 s.h.
GEN -603 Analysis and Simulation Methods	3 s.h.
GEN -604 Test and Evaluation Engineering	3 s.h.

2. Discipline Specialization Courses: The discipline specialization allows the student to further develop his/her ability in a narrow area of engineering. Each specialization must have a minimum of three courses. The primary specializations and the cognizant departments are as follows:

- Civil Engineering: Environmental Systems, Structural Systems, Transportation Systems
- Electrical Engineering: Control Systems, Electrical Power Systems, Micro/Nanoelectronics, Sensor Systems, Software Engineering, Telecommunication Systems

- Mechanical Engineering: Aerospace Systems, Manufacturing Systems, Systems Engineering, Thermal Energy Systems

The student will select courses for his/her specialization from an approved list. Department Advisors may approve other specializations. GEN-590 and GEN-600 may be used as discipline specialization courses. Advisors may also approve appropriate courses from other AAMU departments or courses taken elsewhere.

3. Approved Elective Courses: To allow additional diversity, approved elective courses may be included to complete the course requirements, the number of such electives depending on the number of courses taken in the specialization. These electives may be drawn from other specialty areas or may include approved graduate courses from other AAMU departments or elsewhere as approved by the advisor. In certain cases and as appropriate, advanced courses needed as prerequisites to other program courses may be included.

4. Master's Project: The Master's Project is a significant completion requirement. For this, the student registers in GEN-690 Materiel Engineering Project.

The activity is initiated by a seminar covering the requirements, with an emphasis on reports typical in the engineering profession. The project subject, which must be approved by the student's department advisor, must relate one or more topics from core courses with a detailed topic from a specialty course, providing a state-of-the-art treatment based on available literature. The student then acts independently in developing the report.

Information sources would include journals, trade publications, and government and industry reports. Other than the assessment of this existing information, no original research is required. Source information must be referenced in a professional style. The final report will be reviewed by the department advisor, with a Pass/Fail/Incomplete grade.

Transfer Credit. With the approval of the department advisor, graduate credit may be transferred into this program from study at AAMU and/or other colleges and universities. This would be limited to 12 semester hours if the credit has not been used in completing a graduate program, and 6 semester hours if previously used in completing a graduate degree. All such courses must have been completed with a grade of "B" or better and must have been taken while the student was in a graduate status.

COURSE DESCRIPTIONS

Materiel Engineering

GEN 590 Special Topics – 3 hrs. This course focuses on topics based on modern trends in materiel engineering. The specifics of each course will be identified prior to it being offered.

GEN 600 Special Topics – 3 hrs. This course focuses on topics based on modern trends in materiel engineering. The specifics of each course will be identified prior to it being offered..

GEN 601 Life-Cycle Design Engineering – 3 hrs. This course is intended to provide insight and experience in theory and in practice in dealing with product complexity associated with such design

processes. Topics include contemporary techniques such as product realization process, robust design, design for six-sigma, and design for manufacturability. Also considered are systems architectural principles; system optimization; standardization; and case studies in real-life product design projects. **Prerequisite:** bachelor's degree in engineering or admission to Materiel Engineering graduate program.

GEN 602 Product Assurance Engineering – 3 hrs. This course involves techniques for establishing product specifications, process controls for quality assurance, compatibility analysis, and product reliability and maintainability. Topics include system reliability; confidence intervals-limits; normal and exponential distribution; failure analysis; the Weibull model in life testing; quality control; aging and system reliability; and case studies. **Prerequisite:** bachelor's degree in engineering or admission to Materiel Engineering graduate program; basic knowledge of statistics.

GEN 603 Analysis and Simulation Methods – 3 hrs. The course centers on stochastic search methods for system optimization and the analysis and construction of Monte Carlo simulations. The focus is on issues in algorithm design and mathematical modeling, together with implications for practical implementation. Finite-element analysis is also given major consideration. **Prerequisite:** bachelor's degree in engineering or admission to Materiel Engineering graduate program; capability in computer programming.

GEN 604 Test and Evaluation Engineering – 3 hrs, lecture and laboratory. This course provides an intensive introduction to test methods and evaluation techniques; statistical considerations in measurement uncertainties; experiment planning, designing, debugging, and execution; instrumentation for data acquisition; signal processing; techniques for data analysis and evaluation; methods for hardware verification and validation. **Prerequisite:** bachelor's degree in engineering or admission to Materiel Engineering graduate program; basic knowledge of statistics and electronic instrumentation.

GEN 690 Materiel Engineering Project – 3 hrs. The activity is initiated by a seminar covering the requirements, with an emphasis on reports typical in the engineering profession. The project subject must relate one or more topics from core courses with a detailed topic from a specialty course, providing a state-of-the-art treatment based on available literature. **Prerequisite:** at least 18 graduate semester hours in materiel engineering program.

A number of courses are offered by the three AAMU Engineering Departments for use in fulfilling the Discipline Specialization and/or Approved Elective requirements. Many of these courses are co-listed as undergraduate (400-level) and graduate (500-level). Graduate students taking co-listed courses will be required to complete additional assignments.

Co-listed courses may be used in a graduate program only if the student has not completed the same, or a highly similar, course at the undergraduate level within the past four years (the typical half-life of advanced engineering subjects). If a co-listed course is required in an AAMU undergraduate curriculum, except with approval by the cognizant Graduate Advisor the corresponding 500-level course cannot be used in a Discipline Specialization; these courses, however, may be used in the Approved Electives. Such courses are marked with an asterisk (*).

Conclusions and Future Work

Conclusions

This work documents the research efforts supported by this award involving two faculty members from the Department of Electrical Engineering at Alabama A&M University. The thrust of constituent research projects has been directed towards development, formulation, and implementation of new algorithms with potential applications to automatic fault diagnostics, prognostics, and condition based maintenance. Each year two or more three-member teams of electrical engineering students have participated in research and development activities derived from this effort as part of the required two-semester Senior Design Project course sequence. Research and development activities associated with this award have resulted in advancing state of knowledge in areas of pattern recognition, supervised learning, self organizing clusters, and Bayesian networks. Student participations in research projects associated with this effort have led to senior design projects in areas of importance to industry and government. This has helped the EE Department to equip its BSEE graduates with skill sets that make them better prepared for industrial and government R&D careers as well as pursuit of advanced degrees.

The work in the area of pattern recognition and supervised learning resulted in significant improvements in the Margin Setting algorithms developed previously at AAMU. The new approach utilizes ellipsoidal surfaces in the feature space as class separators. This approach results in production of more robust classifiers with smaller number of classifier filter rounds compared to the original formulation. Examples using simulated and real data show the superiority of this approach. It has been shown that the performance of the ellipsoidal classifier approaches that of the optimal classifier. This was demonstrated with multi class problems using training data obtained from normal distributions with known statistics, where the training algorithm is oblivious to the actual statistical distributions of the training data. Applications of this algorithm include fault diagnostics and prognostics.

The development of a Bayesian Network graph partitioning methodology based upon bisection of the maximum graph diameter was presented. The constraining boundaries were discussed and examined for their roles' in shaping the partitioning scheme. Background information on the graph data structure and traversal methods, and their applicability to BN-graphs was presented. An intuitive method of partitioning was presented. The resultant algorithm, BN_PARTITION, was presented more formally in a pseudo-code style format. Its recursive implementation was analyzed for algorithmic complexity. The resultant complexity is $O(n^{5/3})$. The resultant partitions from the algorithm are of uniform size, (considering the geometric irregularities), and share minimal interfacial zones, which should help reduce communications overhead over arbitrary decompositions.

A theoretical framework and algorithmic methodology for obtaining useful diagnostic and prognostic data from electro-mechanical systems was developed and presented. The

methods were based on vibration and modal analyses of the physical components. Two “real world” models, a PCI circuit card, and an example rotor hub were simulated. Application of the developed techniques proved very useful in identifying FAULTS on the simulated systems.

Two projects involving students were presented. In the first project, the S-Help tool, a Microsoft Visual Basic .Net windows and web application project, was developed. The working functions of the SHELP software lie within a global module file included in the S-Help project. Altogether, there are 26 working functions in the code that perform tasks such as record, fields and item retrieval, text file creation, reading and writing, XML file creation, navigation, data retrieval and parsing. In addition, Macro and shell execution was necessary to communicate with the WINMINE toolkit. There were different algorithms used to parse and filter text and XML files, such as sorting, grouping, searching and recursion. Concepts such as overloading, optional arguments, inheritance, dynamic creation for display controls and referencing was also necessary to carry out the tasks required of the S-Help tool. In the second project, students developed a prototype intelligent sensor to enable fatigue counting. The results obtained from this work is encouraging, and indicate that using fatigue analysis to track component wear by utilizing frequency domain analysis of component vibration profiles may be very useful in characterizing overall system health for use in a CBM environment.

As part of the project, a course in Pattern Recognition was developed within the Department of Electrical Engineering. The course was designed for advanced senior students, and graduate students. It is presently classified as a “Special Topics” course, EE 490 for undergraduate students, and EE 590 for graduate students. A detailed outline of the Pattern Recognition course and a description of the graduate program were presented.

Future Work

For a number of years, only algorithms with relatively low computational complexity, such as some signal and image processing methods have been migrated to embedded devices. (Walsh, 2006) suggests that FPGAs have emerged as a better alternative for signal processing chores than standard general purpose processors, and a number of design methodologies for implementation of DSP (digital signal processing) algorithms in hardware are discussed by (Bancovic, et.al, 2005). However, due to the recent breakthroughs in device logic density, and in spite of the dearth of cohesive end-to-end tool sets, a number of researchers are beginning to transform more complex applications to reconfigurable computing devices. Applications from a variety of fields are being investigated. Bioinformatics pattern matching algorithms are a natural fit with their long linear vector data sets, and embarrassingly parallel comparison operations, and are now commonly being embedded in reconfigurable hardware as shown by (Qui, et.al., 2007) and (Li, et.al., 2007). The latter also demonstrated a 160x speedup of the Smith-Waterman algorithm over traditional processor. From the field of fracture mechanics, an interesting FPGA implementation of a Choleski solver with complexity of $(\frac{1}{3}N^2)$ was detailed in (Dufour, et.al., 2007) claiming a speedup of 16x, their work also included an

FPGA based dynamic simulation of a PMSM (Permanent Magnet Synchronous Motor). However the floating point portion (solution of the inductance matrix) of the simulation was conducted on a typical CPU. (Motuk et.al., 2005) details a 2-dimensional finite difference scheme for solution of the wave equation on an FPGA. This formulation takes advantage of the FPGA architecture by creating a one to one mapping of the network mesh between logic blocks with the natural grid created by the $(i+1, i-1, j+1, j-1)$ indices of the finite difference formulation. This grid is not dissimilar to neighboring update tables in a Bayesian Network graph system.

While there are a number of industrial entities and university researchers working in and defining the areas of reconfigurable device architecture, configuration and support tool development, and reusable IP block definitions, there is not, as yet, a well defined path for implementation of algorithms on FPGA platforms. Despite the complexity of implementation, researchers are beginning to develop FPGA solutions for higher order algorithms, due to the dramatic speedup times that can be attained. Implementations are often vector based, such as DSP and bioinformatics pattern recognition. A few researchers are working on higher level applications, e.g. $O(N^2)$ storage and complexity, pushing the envelope of current FPGA capability. However, simulations of real-time stochastic formulations characterized as pattern recognition, and by large sparse BN systems have not yet been widely adapted to the devices due to their demanding processing and communication requirements. Recent breakthroughs by device manufacturers, as detailed by (Snyder and Williams, 2007) include dramatic increases in logic density coupled with the addition of nanoscale crossbar switching technology layered over the device cells, and offer the promise of sufficient computing and communications capability to host complex algorithms on embedded FPGA platforms in the near future. Innovative numerical and logic design techniques are needed to exploit the capabilities of this new generation of reconfigurable computing devices for use in statistics based simulations, and pattern recognition applications.

Bibliography

- [1] Alvarado, Fernando L., 1992. Secondary Re-Ordering Methods for Parallel Partitioned Sparse Inverse Solutions. Proceedings of the American Control Conference, Publ by American Automatic Control Council, Green Valley, AZ, USA. v 2. pp 1639-1643.
- [2] Askeland, D.R. The Science and Engineering of Materials, 2nd Ed., Boston, MA: PWS-KENT Publishing, 1989, pp. 777-849, 861
- [3] Baeten, J.C.M. and Weijland, W.P., 1990. Process Algebra, 2nd Edition, Cambridge University Press, Ipswich, Suffolk, inc.
- [4] Baldwin, J.F., "Introduction to Bayesian Nets", 2001-2002 lecture notes, <<http://www.enm.bris.ac.uk/teaching/enjfb/emat41100/>>
- [5] Balin, Sidney, 1989. An Object-Oriented Requirements Specification Method, Communications of the ACM, Vol. 32, No. 5 (May) pp. 608-623.
- [6] Bancovic, K., Khalid, M.A.S., Abdel-Raheem, E.. "FPGA-based rapid prototyping of digital signal processing systems." 48th Symposium on Circuits and Systems, 2005. Aug. 7-10, 2005. v. 1, pp. 647-650.
- [7] Bathe, K. Finite Element Procedures, Englewood Cliffs, New Jersey: Prentice Hall, 1996, pp. 785-801
- [8] Burgess, C.J.C., A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery: 2 (2) 955-974, 1998.
- [9] Caulfield, H.J. and Heidary, K., "Exploring Margin Setting for Good Generalization in Multiple Class Discrimination," Journal of Pattern Recognition, Volume 38, Issue 8, 1225-1238, (2005)
- [10] Chlebkova, Janka, 1996. Approximating the Maximally Balanced Connected Partition Problem in Graphs. Information Processing Letters, Vol. 60, No. 5, pp. 225 – 230.
- [11] Chung, T. J. Continuum Mechanics, Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1988.
- [12] Cooper, Gregory; The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks; Artificial Intelligence; vol. 42; pp. 393-405; 1990.
- [13] Cormen, Thomas H., Leiserson, Charles E. and Rivest, Ronald L. 1995. Introduction to Algorithms, 15th Ed., McGraw Hill, New York, pp. 730 – 766.
- [14] Cowell, L.G., Kim, H.J., Humaljoki, T., Berek, C., Kepler, T.B., "Enhanced Evolvability in Immunoglobulin V Genes under Somatic Hypermutation," Journal of Molecular Evolution, Vol. 21, No. 1, 23-26 (1999).
- [15] Duda, R., Hart, P. and Stork, D., Pattern classification (2nd edition), Wiley, New York, 2001.
- [16] Dufour, C., Belanger, J., Abourida, S. and Lapointe, V. "FPGA-Based Real-Time Simulation of Finite-Element Analysis Permanent Magnet Synchronous Machine Drives." 38th Annual IEEE Power Electronics Specialists Conference (PESC '07), Orlando, FL. June 17-21, 2007.
- [17] Fisher, R.A. 1936. The use of multiple measurements in taxonomic problems. Annals of Eugenics, July, 179-188.
- [18] Gelman, A., Carlin, J., Stern, H., Rubin, D.; Bayesian Data Analysis; ISBN: 0412039915; 1st edition, June 1, 1995; CRC Press.
- [19] Genie & Smile: <http://www.sis.pitt.edu/~genie/>
- [20] Graphical Models and Bayesian Networks; <http://www.cs.berkeley.edu/~murphyk/Bayes/>; University of California Berkeley, 2002.
- [21] Guo, H. and Hsu, W.; A Survey of Algorithms for Real-Time Bayesian Network Inference; Laboratory for Knowledge discovery in Databases; Department of Computing and Information Sciences, Kansas State University; 2002.
- [22] Heidary, K. and Caulfield, H.J., "Application of Super-Generalized Matched Filters (SGMF) to Target Classification," Journal of Applied Optics, Volume 44, Issue 1, 47-54 (2005).
- [23] Heidary, K. and Caulfield, H.J., "Discrimination among similar looking, noisy color patches using Margin Setting," Optics Express, Journal of Optical Society of America, Vol. 15, No. 1, 62-75 (2007).
- [24] Heidary, K. and Caulfield, H.J., "Needles in a Haystack: Fast spatial search for targets in similar-looking backgrounds," Submitted to IEEE Transaction on Image Processing (2007).

- [25] Hendrickson, Bruce and Leland, Robert, 1995. A Multilevel Algorithm for Partitioning Graphs. Proceedings of the 1995 ACM/IEEE Supercomputing Conference, Vol. 1, IEEE Los Alamitos, CA pp. 626 – 657.
- [26] Hsu, John S. and Thomas Leonard, Bayesian Methods, Cambridge University Press, 1999
- [27] Ifeachor, E. and Jervis, B. Digital Signal Processing, 2nd Ed., Englewood Cliffs, New Jersey: Prentice Hall, 2002, pp. 111-128
- [28] Jensen, Finn V.; Bayesian Networks and Decision Graphs (Statistics for Engineering and Information Science); ISBN:0387952594 ; 2001; Springer Verlag.
- [29] Jess, J. and Kees, H. “A Data Structure for Parallel L/U Decomposition.” IEEE Transactions on Computers, Vol. C-31, No. 3 March 1982, pp. 231-239.
- [30] John Shawe-Taylor & Nello Cristianini, Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
- [31] Kuon, I., Tessier, R. and Rose, J.. “FPGA Architecture: Survey and Challenges.” Foundations and Trends in Computing, In press. 2008.
- [32] Li, I.T., Shum, W., Truong, K. “160-fold acceleration of the Smith-Waterman algorithm using a field programmable gate array (FPGA).” BMC Bioinformatics, vol. 8, pg. 185, June 2007.
- [33] Liu, Joseph W.H., 1985. Modification of the Minimum Degree Algorithm by Multiple Elimination. ACM Transaction on Mathematical Software, Vol. 11, pp. 141 – 153.
- [34] Looney, C. G. and Liang, L.R., Inference via Fuzzy Belief Networks, <<http://www.cs.unr.edu/~liang/fzbn-isca-final.pdf>>
- [35] Motuk, E. et.al., Implementation of finite difference schemes for the wave equation on FPGA,” IEEE Proc. – ICASSP, pp. III 237-40, 2005.
- [36] Murray, C. Bayesian Belief Networks. 19 Dec. 2003. <http://www.murrayc.com/learning/AI/bbn.shtml>>
- [37] Pandey, N., Singhal, Y. and Parihar, M., Visual Studio.Net All-in-one Desk Reference for Dummies, Wiley Publishing, Inc., 2002
- [38] Pearl, Judea; Causality : Models, Reasoning, and Inference; ISBN:0521773628; 01-Mar-2001; Cambridge University Press.
- [39] Pearl, Judea; Evidential Reasoning Using Stochastic Simulation of Causal Models; Artificial Intelligence; vol. 32; pp. 245-257; 1987.
- [40] Pearl, Judea; Fusion, Propagation, and Structuring in Belief Networks; Artificial Intelligence; vol. 29; pp. 241-288; 1986.
- [41] Qiu, Q., et.al., “Hybrid Architecture for Accelerating DNA Codeword Library Searching,” IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2007. CIBCB apos;07. Volume , Issue , 1-5 April 2007 Page(s):323 – 330
- [42] Schapire, R.E., The strength of weak learnability. Machine Learning, 5(2):197-227, 1990.
- [43] Schuermann, J. Pattern Classification: A Unified View of Statistical and Neural Approaches, Wiley & Sons, 1996.
- [44] Scott, A. “Characterizing System Health Using Modal Analyses.” IEEE Transactions on Instrumentation and Measurement. Accepted June, 2008. 8pp.
- [45] Scott, A. “Data and Process Mapping of Sparse Graph Systems in a Distributed Environment.” Proceedings of SoutheastCon 2008, IEEE. April 2008. pp. 257-258.
- [46] Scott, A. “Electro-Mechanical Diagnostics Prognostics,” IEEE AUTOTESTCON 2007, Baltimore, MD, September 2007, pp. 340-348.
- [47] Simon, H. 1991. Partitioning of Unstructured Problems for Parallel Processing. Computing Systems in Engineering, Vol. 2, No. 3, pp. 135 – 148.
- [48] Sivia, D.S. , Data Analysis: A Bayesian Tutorial; ISBN: 0198518897; July-1996; Oxford Science Publications.
- [49] Snider, G. and Williams, R., “Nano/CMOS architectures using a field-programmable nanowire interconnect,” Nanotechnology **18** pp 1-11, January 2007.
- [50] Sobczyk, K. and Trebick, J. “Fatigue crack growth in random residual stresses,” International Journal of Fatigue, Vol. 26, Issue 11, November 2004, Pages 1179-1187
- [51] Sonntag, R. and van Wylen, G. Introduction to Thermodynamics, Classical and Statistical, 2nd Ed., New York: John Wiley and Sons Publishing, 1982.
- [52] Stach, J., 1995. An Algebraic Method for the Extraction of Concurrent Processing Grains from Object Based Specifications. Ph.D. Dissertation, The Union Institute, Cincinnati, Ohio. inc.

- [53] Tapiador, M., Gomez, J., Siguenza, J.A., Writer Identification Forensic System Based on Support Vector Machines with Connected Components, in *Innovations in Applied Artificial Intelligence*, Vol. 3029, 625-632, (2004)
- [54] Thomson, W.T. Theory of Vibration with Applications, 3rd Ed., Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1988, pp. 257-287, 363-380
- [55] Valiant, L.G., A theory of learnability, *Communications ACM* 27 (1984), 1134-1142.
- [56] Vapnik, V.: *Statistical Learning Theory*, John Wiley and Sons, New York, 1998.
- [57] Vapnik, V.: *The Nature of Statistical Learning Theory*, Springer-Verlog, New York, 1995.
- [58] Walsh, B. "FPGAs Edge Out GPPs for Advanced Signal Processing Apps." *COTS Journal*, <http://www.cotsjournalonline.com/home/article.php?id=100527>, July 2006.
- [59] Whaley, P.W. "Entropy Production during Fatigue as a Criterion for Failure. A Local Theory of Fracture in Engineering Materials," DTIC Publications, ADA151289, Dec. 15, 1984
- [60] WINMINE: Chickering, David Maxwell. 2002. "The WinMine Toolkit", Microsoft, Redmond, WA
- [61] Wu, B., Saxena, A., et.al., "An Approach to Fault Diagnosis of Helicopter Planetay Gears," *IEEE Proceedings of AUTOTESTCON 2004*, September 2004, pp. 475-481
- [62] Xilinx, Virtex 5 multi-platform FPGA. In http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex5/index.htm, 2008.
- [63] Yannakakis, M., 1981. Computing the Minimum Fill-In is NP-Complete. *SIAM Journal of Algorithms and Discrete Methods*. Vol. 2, No. 1, pp. 77 – 79, March.